НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра
з напряму підготовки 6.050103 «Програмна інженерія»
на тему: Веб-ресурс «Мапа подій»

Виконав:
студент IV курсу, групи КП-52
Самохваленко Станіслав Ігорович _____
Керівник:
Ст. викладач кафедри ПЗКС,
Бухтіяров Ю.В. _____
Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____
Рецензент:
Доцент кафедри ЕІ ФЕЛ, к.т.н.,
Вунтесмері Ю.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Факультет прикладної математики

Національний технічний університет України

“Київський політехнічний інститут

імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050103 “Програмна інженерія”

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

“___” _____ 2018 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Самохваленку Станіславу Ігоровичу

1. Тема проекту веб-ресурс «Мапа подій», керівник проекту Бухтіяров Юрій Вікторович, старший викладач, затверджені наказом по університету від “22” травня 2019 року №1331-С

2. Термін подання студентом проекту: “20” червня 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Зміст пояснювальної записки:
   – розробити загальну структуру клієнтської частини сервісу;
   – визначити організацію інформації на кожній сторінці сервісу;
   – виконати програмну реалізацію веб-сервісу відповідно до вимог технічного завдання;
   – виконати тестування веб-сервісу.
5. Перелік обов’язкового ілюстративного матеріалу:

– діаграма прецедентів (креслення);
– ERD діаграма (креслення);
– структура клієнтського веб-додатку Angular (плакат);
– архітектура розробленого веб-додатку (плакат).

6. Консультанти:

Питання	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання: “31” жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання етапів	Примітка
1.	Вивчення літератури за тематикою проекту	29.11.2018	
2.	Розроблення та узгодження технічного завдання	29.11.2018	
3.	Розроблення структури інформаційної системи	16.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	18.01.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	30.01.2019	
6.	Програмна реалізація інформаційної системи	07.04.2019	
7.	Тестування інформаційної системи	27.04.2019	
8.	Підготовка матеріалів третього розділу дипломного проекту	05.05.2019	
9.	Підготовка матеріалів четвертого розділу дипломного проекту	21.05.2019	
10.	Підготовка графічної частини дипломного проекту	01.06.2019	
11.	Оформлення документації дипломного проекту	08.06.2019	

Студент _____ Самохваленко С.І.

Керівник проекту _____ Бухтіяров Ю.В.

Самохваленко Станіслав Ігорович
(прізвище, ім'я, по батькові студента)

ВЕБ-РЕСУРС «МАПА ПОДІЙ»

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмного забезпечення для веб-ресурсу, що являє собою мапу подій.

Інтерактивна інформаційна система являє собою односторінковий web-додаток. Для не зареєстрованого користувача доступні функції перегляду подій у певному регіоні та перегляд деталізованої інформації про подію. Для зареєстрованого користувача доступні функції перегляду, пошуку, фільтрації, додавання, редагування та видалення подій, а також їх додавання у список улюблених. Після отримання дозволу модератора сайту користувач може додати власну подію на мапу. Після проходження авторизації користувач отримує змогу шукати та фільтрувати події за такими критеріями: назва, локація проведення, дата проведення.

У даному дипломному проекті розроблено: архітектуру інформаційної системи, алгоритм авторизації, модулі геокодингу та відображення інформації на інтерактивній мапі, а також графічні елементи та дизайн web-сторінок.

Samokhvalenko Stanislav

(student first name, last name)

WEB-RESOURCE «EVENTS MAP»

ABSTRACT

This diploma project is dedicated to creating a software for a web-application which represents a map with events.

The interactive information system is a single-page application. An unauthorized user can view general and detailed information about events based on a region search. For an authorized user there is an access to viewing, searching, filtering, editing and deleting events as well as add them to favorite lists. After moderation user can add his event to a map. After going through authorization procedure user can search and filter events by the following criteria's: title, location and date.

During this project it was developed the following: architecture of the information system, algorithm of authorization, modules for geocoding and displaying information on an interactive map as well as graphic and design elements for web-pages.

Студент _____ _____

(*підпис*) (*прізвище, ім'я, по батькові*)

Науковий керівник _____ _____

(*підпис*) (*посада, вчене звання, науковий ступінь, прізвище, ініціали*)

ДП.045440-01-90 Веб-ресурс «Мапа подій.» Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-ресурс «Мапа подій».	4	
	Технічне завдання		
ДП.045440-03-81	Веб-ресурс «Мапа подій».	51	
	Пояснювальна записка		
ДП.045440-04-51	Веб-ресурс «Мапа подій».	4	
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-ресурс «Мапа подій».	4	
	Керівництво користувача		
ДП.045440-06-99	Веб-ресурс «Мапа подій».	1	
	Діаграма прецедентів.		
	Функціональність		
	програмних засобів		
ДП.045440-07-99	Веб-ресурс «Мапа подій».	1	
	ERD діаграма. Структура		
	бази даних		
ДП.045440-08-98	Веб-ресурс «Мапа подій».	1	
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2018 р.

ВЕБ-РЕСУРС «МАПА ПОДІЙ»

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В.Онай

Виконавець:

_____ С.І. Самохваленко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-ресурс «Мапа подій».

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості інформаційного ресурсу в мережі Інтернет для пошуку і перегляду подій у культурно-розважальному просторі.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Система повинна забезпечувати такі основні функції:

1) збереження подій з прив'язкою до місця і часу;
2) доступ користувачів до перегляду, створення і редагування подій;
3) інтеграція з сервісом Google Maps для прив'язки подій до місця їх проведення;
4) швидкий та зручний пошук подій з використанням пошукових фільтрів;

Розробку виконати на сучасному фреймворку з використанням картографічних сервісів.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Функціональність програмних засобів. Діаграма прецедентів»;
   - «Структура бази даних. ERD діаграма».

6. ЕТАПИ ПРОЕКТУВАННЯ

Ознайомлення з літературою за предметною областю 15.10.2018
Розробка технічного завдання 15.11.2018
Розробка структури web-сервісу 30.11.2018
Розробка дизайну сторінок 20.12.2018
Програмна реалізація web-сервісу 08.02.2019
Тестування web-сервісу 25.03.2019
Підготовка матеріалів текстової частини проекту 09.04.2019
Підготовка матеріалів графічної частини проекту 07.05.2019
Оформлення програмної документації 12.06.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ І.А.Дичка

“___”_____ 2019 р.

ВЕБ-РЕСУРС «МАПА ПОДІЙ»

Пояснювальна записка

ПЗКС.045440-03-81

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.В. Бухтіяров

Виконавець:

Нормоконтроль: _____С.І. Самохваленко

_____ М.В. Онай

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

GPS (Global Positioning System) – сукупність радіоелектронних засобів, що дозволяє визначати положення та швидкість руху об'єкта на поверхні Землі або в атмосфері.

CLI (command-line interface, інтерфейс командного рядка) — різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків, які являють собою команди.

NoSQL (від англ. no only SQL) – база даних, що забезпечує інший механізм зберігання та видобування даних, ніж звичайний підхід таблиць-відношень в реляційних базах даних.

HTTP (від англ. Hypertext Transfer Protocol) – протокол передачі гіпер-текстових документів, що використовується в комп’ютерних мережах.

ACID (від англ. Atomicity, Consistency, Isolation, Durability) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

OAuth (скорочення від англ. Open Authorization) – це відкритий стандарт авторизації, який дозволяє користувачам відкривати доступ до своїх приватних даних (фотографій, відео, списки контактів), що зберігаються на одному сайті, іншому сайту, без необхідності вводу імені користувача або паролю.

OpenID – це децентралізована система єдиного входу, котра дозволяє використовувати один обліковий запис (ім'я користувача і пароль) на великій кількості сайтів.

Фреймворк (Framework, каркас, платформа) – інфраструктура програмних рішень, що полегшує розроблення складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою.

API (від англ. Application Programming Interface) – прикладний програмний

інтерфейс, набір визначень взаємодії різнотипного програмного забезпечення).

Back end – серверна частина додатку.

Front end – це інтерфейс для взаємодії між користувачем і back end. Front end та back end можуть бути розподілені між однією або кількома системами.

REST (від англ. Representational State Transfer, «передача репрезентативного стану») – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

JSON (від англ. JavaScript Object Notation) – це текстовий формат обміну даними між комп’ютерами.

ВСТУП

У сучасному світі засилля даних різних видів можна спостерігати неозброєним оком. Люди кожного дня зазнають впливу величезного хаотичного потоку контенту, велика частина якого не несе жодної користі кінцевому отримувачу. Саме тому стають популярними так звані онлайн-сервіси агрегатори, які дозволяють людям отримати саме те, що їм потрібно, відсікаючи величезні обсяги непотрібних даних. Зважаючи на це, зростає час уведення, обробки та аналізу цих даних. Сфера розваг не є виключенням. На даний час розроблена неймовірна кількість найрізноманітніших програмних застосунків для обслуговування людських потреб.

Щодня ми маємо змогу бачити величезний наплив реклами, промо-акцій та інших засобів, якими виробники програмного забезпечення намагаються збільшити кількість користувачів своїх програм. Вони відрізняються функціоналом, вартістю, сферою обслуговування та колом користувачів, що зацікавлені у користуванні цією програмою. Одними з найбільш відомих агрегаторів контенту в Україні є ukr.net - агрегує новини з величезної кількості інших веб-ресурсів, prom.ua - торгова платформа для бізнесу, де розміщують свої товари десятки а то і сотні інтернет-магазинів, hotline.ua – агрегатор даних про інтернет магазини, спеціалізується на товарах та порівнянні їх цін на різних платформах.

Попри те, що з кожним роком агрегаторів стає все більше, деякі області людського життя вони все ще не покрили. Веб-ресурс та програмна частина застосунку, описаного в цій роботі, покриває одну з цих областей – події. На сьогоднішній день існує невелика кількість агрегаторів подій, що допомагають користувачам отримати повний набір інформації про подію, яку вони хочуть відвідати, а таких зараз величезне різноманіття: концерти, вечірки, лекції, майстер-класи і ще безліч різних можливостей для цікавого проведення вільного часу. Та все ж існуючі програмні рішення далеко не повністю задовольняють потреби користувачів, наприклад мають не надто зручний для

користувача інтерфейс, забирають його увагу непотрібними даними або ж не надають інформацію у зручному для аналізу вигляді. Також на даний момент не існує єдиного агрегатора подій з усього світу з можливістю зручного їх перегляду.

Основними відмінностями розробленої системи є можливість переглядати події по всій території нашої планети за допомогою інтерфейсу-мапи, на які користувач зможе відфільтрувати події за потрібними параметрами, як от регіон, час проведення, ціна та вид події. Саме це і послугувало причиною розробки даного програмного застосунку.

Після впровадження системи користувачі матимуть можливість введення та редагування даних про власні події, а також перегляду і фільтрування подій за потрібними параметрами.

1. ОГЛЯД НАЯВНИХ РІШЕНЬ

1.1. Загальні відомості про веб-ресурси агрегатори та їх розвиток

Діяльність агрегаторів різних типів даних тісно пов’язана з розробкою та постійним покращенням функціоналу додатку для постійного притоку нових користувачів. Вміння збирати дані, надавати їх користувачам у найзручнішому вигляді, з постійно актуальною інформацією дозволяє на вищому якісному рівні підходити до розробки та просування власного агрегатора.

Якщо ви користуєтесь інтернетом давно, то пам’ятаєте епоху, коли в магазинах продавались паперові довідники і журнали, які містили каталоги сайтів. Також були і сайти-довідники – «жовті сторінки інтернету». Потім в українському та російськомовному сегменті інтернету набув популярності “Рамблер”, а його вже наздогнав і обігнав “Яндекс”, з'явився російський “Google”. Але ще до пошукачів були агрегатори, сайти-каталоги. І вони з часом не зникли, лише підлаштувались під потреби своєї аудиторії. Хоча деякі з них майже не міняли дизайн за останні 15-20 років.

Сайт-агрегатор («мешап», «Mash-up», змішувати) – веб-додаток або інтернет-сайт, що об'єднує дані з декількох джерел в один з єдиним призначеним для користувача інтерфейсом; наприклад, сайти-агрегатори відгуків різних кіно- чи інших критиків (Rotten Tomatoes і Metacritic), сайти для покупки авіаквитків з можливістю вибору найбільш зручної пропозиції, сайти для бронювання готелю.

На території сучасної України використовуються різні програмні застосунки-агрегатори. Деякі з існуючих програмних рішень є загальновідомими у суспільстві, як от Uber чи Hotline. Але всі існуючі аналоги мають свої переваги та недоліки. Далі наведено загальні відомості про деякі найбільш поширені веб-сервіси агрегатори. Через відсутність достатньої кількості аналогічних проектів будуть розглянуті веб-ресурси агрегатори різної тематики, такі як Ukr.net, Hotline.

1.2. Огляд наявних програмних рішень

1.2.1. Агрегатор даних про товари інтернет-магазинів Hotline

Hotline (хотлайн) – український онлайн-сервіс і агрегатор для вибору товарів і порівняння цін. У 2011 році на Хотлайн покупцям було доступно більше 2 млн. пропозицій від усіх провідних і активно працюючих інтернет-магазинів України - більше 1400 магазинів. Станом на липень 2012 року на сайті розміщувалось близько 2 млн пропозицій від 1700 інтернет-магазинів.

На даний момент середня кількість відвідувань сайту на місяць складає більше 8 мільйонів, а кількість щоденних переглядів – більше трьохсот тисяч.

Hotline (хотлайн) – програмний комплекс, що збирає дані про товари, магазини, ціни на товари та відгуки користувачів, що дозволяє при потребі переглянути усю необхідну інформацію про товар чи магазин, порівняти її у зручному інтерфейсі, що надає можливість також відсортувати дані за потрібними параметрами, як от перегляд одного й того ж товару в різних магазинах з сортуванням за ціною або рейтингом.

Система включає в себе збір даних з різних інтернет-магазинів, що є вигідним і для продавців, які отримують можливість збільшити кількість користувачів через відображення своїх товарів у списках на hotline. Серед іншого, є можливість пошуку товару за кодом товару, який є єдиним для усіх магазинів, що унеможливлює випадок, коли в порівняння попали схожі, але різні товари від магазинів. Також реалізовано можливість фільтрації за унікальними для кожного товару параметрами, що дозволяє чи не найкращим чином звузити вибірку відображених користувачі товарів та надати найбільш потрібну інформацію без засмічення уваги користувача непотрібними йому даними. Наразі в Україні він є найбільш популярною платформою для порівняння цін на товари, що автоматично приваблює продавців і спонукає їх аналізувати дані з hotline і встановлювати відповідну цінову політику.

Цей веб-ресурс надає широкий функціонал не тільки для рядових користувачів, а й для магазинів, які отримують можливість просувати свої товари завдяки діяльності

hotline, нижче будуть наведені можливості для продаців.

Реклама товарів: система відображає користувачам рекламу товарів, яку оплачує інтернет-магазин. Усі цінові списки відображені на прямо сайті, що дозволяє зручно планувати та змінювати рекламну кампанію за допомогою Хотлайн.

Також реалізований аукціон – просування товару та визначення його ціни на льоту під час перегляду товарів користувачем, під час якої також змінюється вартість одного переходу з hotline на власне сайт магазину.

Були реалізовані і можливості, які дозволяють зручніше взаємодіяти користувачам з магазинами і приносять користь обом.

Hotline Checkout - це сервіс, що дозволяє користувачам hotline.ua розрахуватися платіжними картами за товари, представлені на hotline.ua інтернет-магазинами. В цьому випадку hotline виступає як "посередник" при оформленні замовлення і проведенні платежу. Магазин і hotline.ua стають партнерами: hotline.ua виконує роль агента і адміністратора сервісу, а магазин - продавця і постачальника.

Hotline пропонує усім фірмам які торгують вроздріб, стати учасником системи Hotline Checkout. Працюючи з Checkout, магазин може «вивантажувати» в Систему весь свій товарний асортимент або його частину і продавати ці товари користувачам hotline.ua з передоплатою платіжною карткою.

Крім того, Checkout має широкий функціонал для інтернет-магазину по роботі з покупцем: налаштування параметрів доставки, скасування замовлення та повернення грошових коштів, автоматизація цих процесів через API.

Основні недоліки:

– банерна реклама;
– обов’язкова плата за перехід до інтернет магазину для власника магазину.

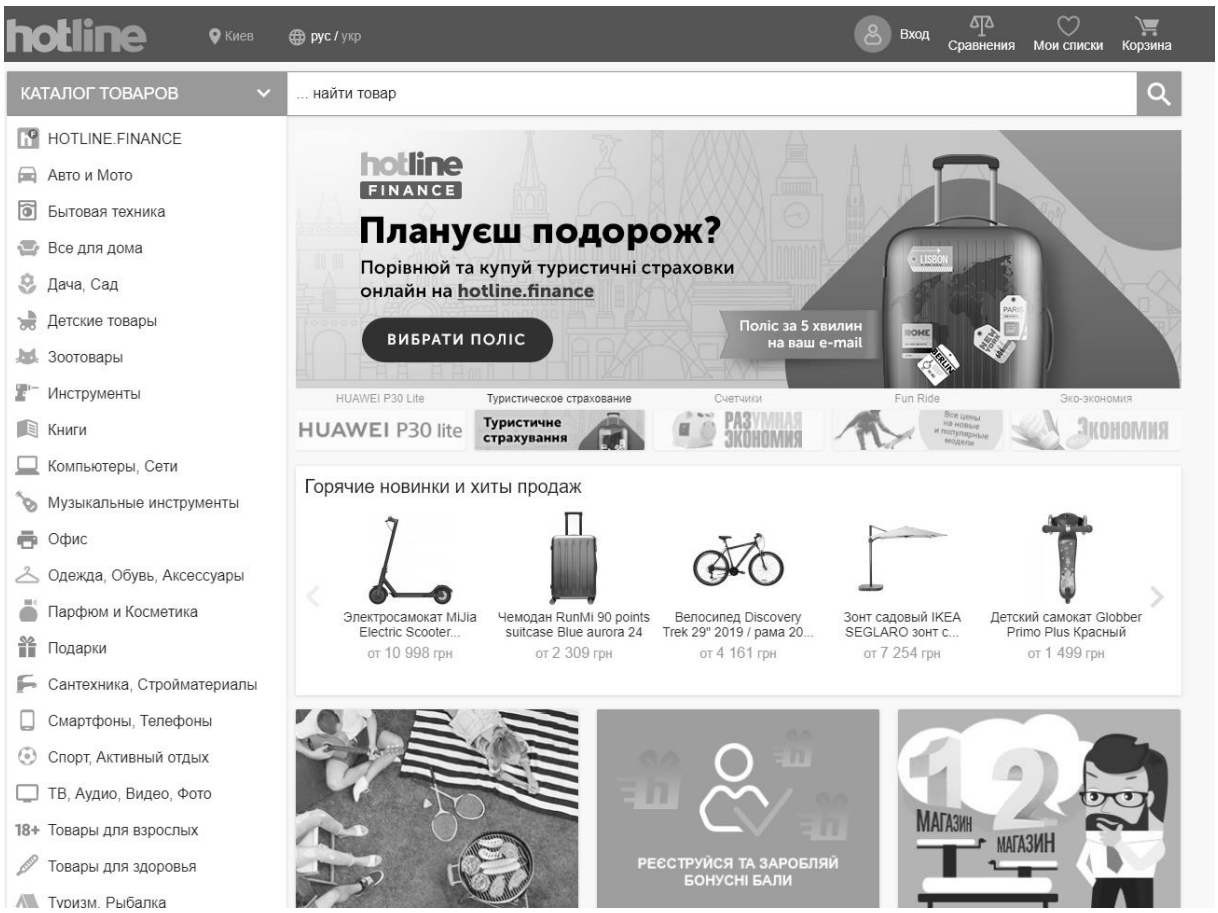


Рис. 1. Графічний інтерфейс користувача веб-сайту hotline.ua

1.2.2. Ukr.net

Ukr.net – український веб-портал, в минулому – інтернет-провайдер.

На початок 2017 року Ukr.net - найбільш популярний український веб-ресурс агрегатор новин. Щодня в новинну стрічку до порталу Ukr.net поступає декілька десятків тисяч новин з більш ніж тисячі різних веб-ресурсів з усієї України. Серед них і широко відомі новинні сайти з широкою аудиторією, інформаційні агентства різного масштабу, як українські, так і міжнародні, а також веб-ресурси різних ЗМІ, онлайн-ресурси певних регіонів.

Новини, що відображаються тематичних рубриках на ресурсі, оновлюються щохвилини. Представлені рубрики: політика, економіка, події, суспільство, технології, наука, авто, спорт, здоров'я, шоу-бізнес, за кордоном, цікавинки, фоторепортаж, відео,

рубрики з новинами окремих регіонів України. На головній сторінці в новинній стрічці UKR.NET відображаються найбільш свіжі та актуальні новини, що оновлюються за допомогою алгоритмів.

Веб-ресурс ukr.net не додає та не створює новини чи статті сам по собі та не формує контент. Список новин на веб-порталі ukr.net формується програмними засобами ресурсу автоматично та є сукупністю гіпертекстових посилань на сторінки, що містять новини, опубліковані на різних веб-ресурсах України та світу.

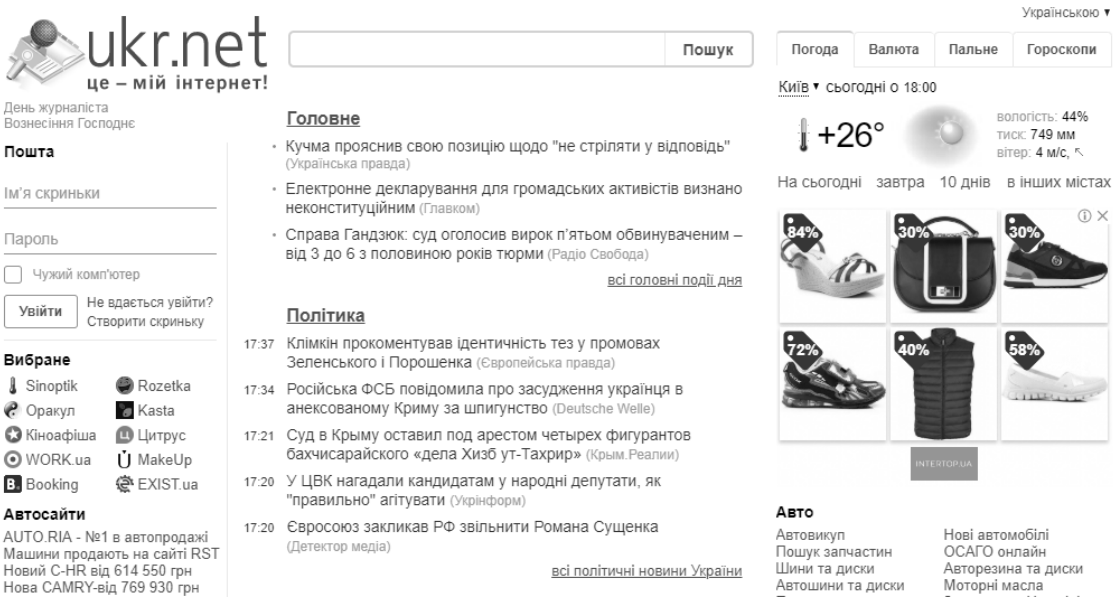


Рис. 2. Графічний інтерфейс користувача сайту ukr.net

Посилання збираються, аналізуються та зберігаються до однієї з релевантних рубрик, в кожній з яких вони групуються відповідно до теми новини чи статті. Вони відображаються на сайті незалежно від того, яку політичну позицію або ідеологію підтримує інтернет-видання, яке їх опублікувало. Це дає змогу користувачу отримати можливість ознайомитися і вибрати посилання на новини та статті, що надають огляд з різних точок зору на потрібну тематику. Користувач має змогу самостійно вибирати цікаві йому теми і джерела, які опублікували відповідну статтю. Для цього йому

необхідно відкрити групу новин, що відносяться до потрібної йому теми та натиснути на релевантне посилання, після чого буде здійснено перехід на потрібний сайт.

Основними недоліками даного ресурсу є:

– відсутність власного пошуку по сайту;
– відсутність фільтрації за звичайними для новин параметрами: дата, час, заголовок.

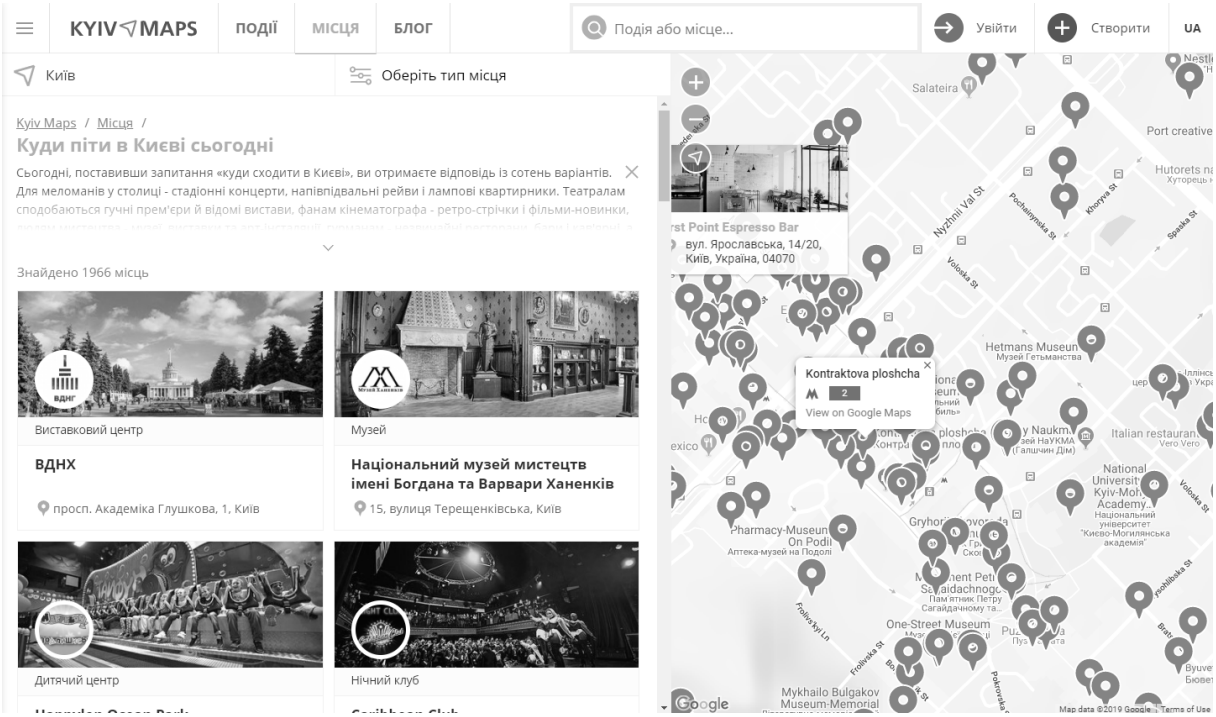
1.2.3. Kyiv maps

Портал Києва, який надає його жителям та гостям актуальну інформацію про найцікавіші та популярні місця, анонси та події столиці, а також довідкові дані про міську інфраструктуру. Ресурс використовує відкритий функціонал карт від Google. На портал додано інформацію про міську інфраструктуру районів Києва.

Веб-ресурс є відкритою для користувачів та безкоштовною онлайн платформою для організаторів різноманітних подій, івентів, концертів, вечірок та будь-яких інших видів заходів. Він надає можливість переглядати та користуватись інформацією не лише про події, а й про місця столиці, наприклад бари, клуби, ресторани, музеї, кінотеатри, парки та торгово-розважальні центри. У власників локацій є можливість додати власну локацію і просувати її на веб-ресурсі.

Користувачам онлайн платформи надається можливість вести свій власний блог, публікуючи статті від власного імені, які після цього стають доступними іншим користувачам. Найчастіше це інтерв’ю чи огляди, які відносяться до нових місць чи подій, що проводяться чи будуть проводитись у Києві.

Рис. 3.



Графічний інтерфейс користувача веб-сайту Kyiv Maps

Веб-сервіс дозволяє переглядати події Києва на карті з маркерами, кожен з яких відповідає певній події. На сайті реалізований пошук по декількох категоріях: події, місця, блоги. Шукати події можна і з допомогою системи фільтрації за місцем проведення, датою та типом події. Місця ж можна відфільтрувати виключно за типом або локацією.

Основними недоліками даного сервісу є:

– обмеженість: інформація на платформі стосується виключно Києва;
– величезна кількість маркерів на мапі, що ускладнює взаємодію користувача з контентом;
– одноманітність кольору, розміру та форми маркерів, що ускладнює їх перегляд користувачем.

1.2.4. Facebook local

Мобільний додаток від Facebook, який дозволяє переглядати та вибирати різні події, місця, заходи у певній місцевості. Рекомендації та відображувані події

підбираються додатком на основі даних про підписки та уподобання вас і ваших друзів у соціальній мережі, а також на основі даних про уже відвідані вами події, тощо.

Додаток містить інтерактивну карту, на якій користувач може переглянути події чи локації в різних регіонах, а також скористатись системою пошуку та фільтрації даних для відображення більш релевантного контенту. Користувач має змогу фільтрувати дані за такими критеріями, як час, місце проведення, тип заходу чи категорія локації.

Інтерфейс мобільного додатку також дозволяє швидко знаходити квитки на потрібну подію чи місце, поширювати інформацію серед друзів за допомогою соціальних мереж, месенджерів чи власне додатку, а також синхронізувати події та плани з календарем на мобільному телефоні.

Основні недоліки:

– відносно невелика кількість користувачів та контенту;
– відсутність онлайн версії додатку;
– інтерфейс не підтримує достатню кількість мов;
– засилля одноманітних маркерів.

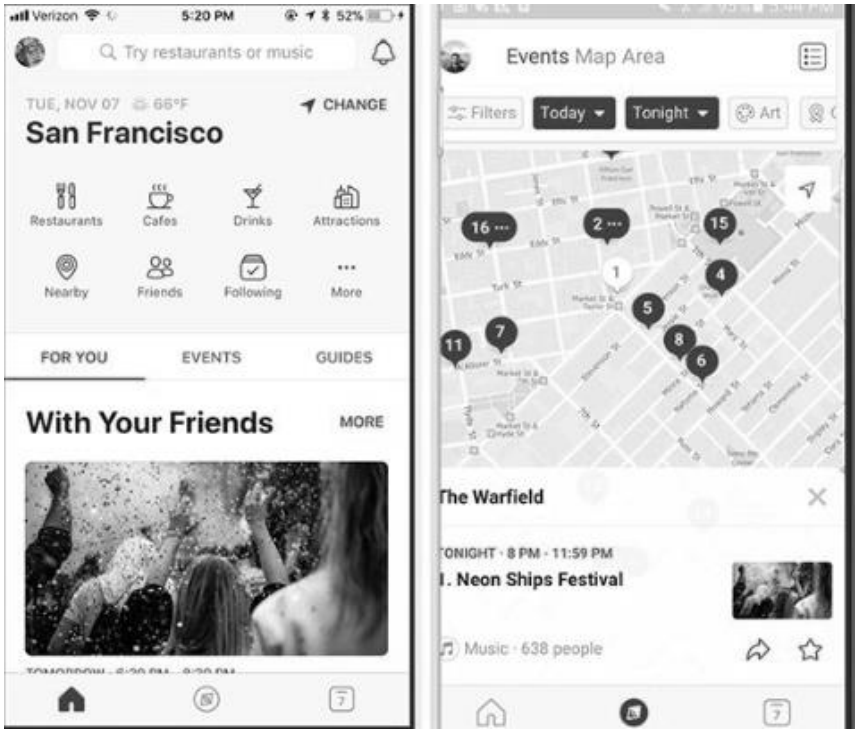


Рис. 4. Графічний інтерфейс користувача додатку Facebook Local

1.3. Висновки

У даному підрозділі були розглянуті веб-ресурси або ж мобільні додатки, які тим чи іншим чином збирають, обробляють і показують користувачеві інформацію з різних джерел. Отримана інформація була систематизована з метою використання в подальшій роботі для визначення вимог до програмної системи розроблюваного додатку. На основі дослідження даних програмних систем можна зробити висновок про їхні особливості, основні переваги та недоліки.

Отже, можна зробити висновок, що існуючі на даний момент в Україні та світі агрегатори інформації про події та місця не задовольняють усі потреби користувачів, мають не достатньо розвинений та зручний інтерфейс і велику кількість вагомих недоліків, що погіршують досвід користування. Існуючі аналоги розвиваються повільно або ж взагалі не розвиваються через недостатню кількість користувачів та непродуманий інтерфейс.

Основними перевагами розглянутих програмних засобів є:

- простота роботи;
- збір актуальної та потрібної інформації;
- велика кількість вже отриманих даних;
- звичний інтерфейс;
- швидка обробка даних.

Основними недоліками українських аналогів є:

- покриття додатками чи веб-ресурсами лише частини території України;
- відсутність мобільних версій проектів, що значно ускладнює роботу з ними;
- незручний інтерфейс;
- банерна реклама;
- низька якість та кількість контенту.

Основними недоліками розглянутих міжнародних програмних засобів є:

- відсутність підтримку україномовного інтерфейсу;

– незручний інтерфейс;
– покриття невеликої кількості країн чи регіонів;
– низький рівень структурованості даних;
– величезна кількість однакових маркерів на мапах, в яких важко орієнтуватись.

На основі наведених переваг та недоліків існуючих рішень були сформульовані наступні вимоги до системи, що розроблюється:

– повинна відображати інтерактивну карту подій та локацій в усьому світі;
– маркери повинні мати чіткі відмінності у зовнішньому вигляді залежно від типу того, що вони відображають;
– користувач повинен мати змогу додавати власні події та місця;
– мапа має регулярно оновлюватись та відображати актуальні дані;
– інтерфейс повинен бути простим та зручним;
– одночасно на мапі повинна відображатись лише невелика частина маркерів;
– користувач повинен мати змогу відфільтрувати контент за будь-якою з характеристик;
– повинна надавати користувачам можливість додавати власні коментарі, фото, позначки на карту.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування

Мовою програмування було вибрано стандартну для таких задач мову JavaScript.

2.1.1. Огляд мови програмування Javascript

JavaScript – прототипна, об'єктно-орієнтована та динамічна мова програмування. Підтримує імперативний, об'єктно-орієнтований та функціональний стилі. Являється реалізацією мови ECMAScript.

Найбільш широке застосування мова отримала в браузерах як основний інструмент для створення сценаріїв для додавання інтерактивних елементів веб-сторінок на стороні клієнта, а також надає можливість взаємодії з користувачем, керування браузером, асинхронного обміну даних з сервером, зміни структури та зовнішнього вигляду веб-сторінки.

Основними архітектурними рисами цієї мови є: автоматичне керування пам'яттю, динамічна типізація, прототипне програмування, використання функцій як об'єктів першого класу, прототипне наслідування.

JavaScript розроблялась під впливом багатьох мов програмування. Під час розробки однією з цілей було розробити інструмент схожим на Java і разом з цим простим для використання та розуміння людей, які не є програмістами. Мова JavaScript не знаходиться у власності будь-яких компаній чи організацій, що відрізняє її від ряду інших мов програмування, які використовуються у веб-розробці.

Javascript використовується для широкого кола задач, і для багатьох має уже готові фреймворки для розробки:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-додатків (React, AngularJS, Vue.js);
- програмування на стороні сервера (Node.js);

– стаціонарних застосунків (Electron, NW.js);
– мобільних додатків (React Native, Cordova);
– сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
– всередині PDF-документів.

Три ключові елементи об'єдналися в технології мови Javascript:

– робота з об’єктно-орієнтованістю – розробник напряму взаємодіє з певними об’єктами.
– динамічна типізація – тип даних не потрібно вказувати, він визначається компілятором.
– найчастіше користувач буде працювати з функціями, які виступлять у вигляді об'єктів базового рівня. Дана особливість робить мову веб-програмування дуже схожим на інші скриптові мови - Lisp або Haskell.
– головна особливість полягає в автоматизованій очищення пам'яті. Така функція повністю відсутня в інших мовах, як C# або Java. Завдяки цій можливості, працювати з мовою особливо початківцю буде простіше.

Основними перевагами мови Javascript є:

– велика кількість можливостей для вирішення найрізноманітніших завдань. Гнучкість мови дозволяє використовувати безліч шаблонів програмування стосовно до конкретних умов;
– популярність JavaScript відкриває перед програмістом чимала кількість готових бібліотек, які дозволяють значно спростити написання коду;
– величезна спільнота розробників, завдяки чому можна отримати своєчасну та актуальну допомогу чи інформацію залежно від потреб.

2.1. Вибір фреймворку для розроблення клієнтської сторони додатку

2.1.1. Angular

Angular (відомий також як Angular 2 або Angular 2+, що означає його кардинальну відмінність від першої версії фреймворку) – розроблений Google та спільнотою розробників front-end фреймворк. Має відкритий код, написаний на TypeScript – мові програмування, що розширює можливості JavaScript.

Angular 2 отримав свою назву від команди розробників, після того як вони практично з нуля створили нову версію даного фреймворку. Вони також вирішили змінити назву його перший версій на AngularJS, а версії 2+ називати просто Angular задля наголошення на разючій різниці між версіями.

Важливою особливістю цього фреймворку є Angular CLI – інтерфейс командного рядка для керування додатками, написаними на Angular. Він значно спрощує створення, редагування та оновлення як самого додатку, так і усіх побічних підключених до нього бібліотек.

Фреймворк використовує як основний концепт ієрархічну структуру компонентів, замість області видимості та контролерів. Це дозволяє поступово та в уніфікований спосіб будувати структуру додатку та розділяти його на модулі.

Основна частина його функціоналу виділена в модулі, які можна підключати до додатку. Розробник має змогу створювати власні модулі зі складною ієрархією та взаємозв’язками.

Завдяки використанню TypeScript розробнику відкриваються додаткові можливості, такі як:

– класи, що дозволяють використовувати стандарти на принципи об’єктно-орієнтованого програмування;
– строга типізація для кращого контролю над даними в додатку та виявленню багів на рівні компіляції;

– узагальнене програмування – побудова функцій, які працюють в один і той же спосіб з різними типами даних;
– динамічне завантаження;
– асинхронна компіляція шаблонів;
– використання директив та компонентів – директива, що містить шаблон;
– використання ітеративних функцій зворотнього виклику.

Важливою особливістю даного фреймворку є те, що кожна його наступна версія планується як зворотньо сумісна з попередньою. В останніх версіях оновлення не потрібно більше проводити вручну, воно можливе за допомогою інтерфейсу командного рядка. Завдяки цьому полегшується підтримка довгострокових та великих проектів та своєчасне їх оновлення.

2.2. Вибір системи керування базами даних

2.2.1. Firebase

Firebase – хмарна база даних типу NoSQL для додатків, які потребують оновлення даних в реальному часі, яка дозволяє розробникам зберігати, оновлювати і синхронізувати дані між декількома клієнтами. Компанія була придбана Google у жовтні 2014 року.

Firebase NoSQL надає базу даних для додатків реального часу в якості сервісу. Ця служба надає API для розробників, дозволяє синхронізувати дані і зберігати їх в хмарі Firebase. Функція синхронізації RTDB надає клієнту всі відсутні файли після підключення відновлення, що робить його дуже цінним для стабільної роботи додатка. Компанія також взяла до уваги можливість інтеграції з Android, IOS, JavaScript, Java, Objective-C і додатками Node.js. Для шифрування даних розробники можуть скористатися API, що надається також Firebase.

Використання Firebase SDK для зберігання і масштабування даних забезпечує

повну безпеку і надійність. Інформація може бути відправлена безпосередньо від клієнта до Firebase Cloud, різні параметри конфіденційності можна регулювати, щоб обмежити доступ до певних груп або файлів. Firebase підтримується Google Cloud, тож може запропонувати розробникам практично нескінченний простір зберігання даних [12].

База даних реального часу Firebase є безсхемною базою даних, в якій дані зберігаються в форматі JSON. Це необхідно враховувати при розробці схеми бази даних так, щоб дані були доступні легко, уникаючи вкладеності дочірніх вузлів. JSON дані мають форму дерева, тож кожна одиниця даних може містити ще декілька одиниць даних, які в свою чергу також можуть містити інші одиниці даних. Використання JSON дозволяє взагалі не структурувати дані, хоча це й не рекомендується. Firebase дозволяє вкладувати дані до 32 рівнів. Це дозволяє мати мільйони даних, які містять мільйони даних.

Більшість баз даних вимагають HTTP викликів для отримання і синхронізації даних. Більшість баз даних надають дані тільки, якщо користувач дає запит на їх отримання. Якщо розробник використовує Firebase, з'єднання відбувається за допомогою WebSocket. WebSockets є значно швидшими за HTTP. Всі дані синхронізуються автоматично через один WebSocket так швидко, як клієнтська мережа може переносити їх.

Оскільки дані зберігаються в хмарі, а не локально на пристрої користувача, це дозволяє користувачу мати доступ до даних з будь-якого пристрою, якщо він авторизується за своїм акаунтом. Це також означає, що додаток потребує інтернет з'єднання, а дані мають бути однозначно пов'язані з користувачем.

Основними перевагами Firebase є:

– сховище JSON означає відсутність бар'єру між даними і об'єктами;
– легка серіалізація стану додатку;
– легке налаштування для використання у додатку;

– легкий доступ до даних, файлів, авторизації тощо;
– не потрібно налаштовувати серверну інфраструктуру для роботи додатку;
– великий потенціал для зберігання великих масивів даних;
– є базою даних в реальному часі;
– висока надійність та безпека;
– легко інтегрується з іншими технологіями Google.

Основними недоліками Firebase є:

– дуже обмежені можливості для запитів і індексування;
– відсутня можливість агрегації;
– відсутній map reduce;
– немає можливості створити запит для перегляду списку користувачів чи збережених файлів.

2.2.2. SQLite

SQLite – є полегшеною версія реляційної системи керування базами даних SQL. У ній реалізована велика кількість можливостей стандарту SQL-92, а розроблена вона у вигляді бібліотеки. Дозволяється використовувати сирцевий код цієї системи у зв'язку з тим, що він знаходиться у відкритому доступі і розповсюджується за відкритою ліцензією. Кожен розробник при бажанні має змогу безоплатно користуватись усіма її можливостями без жодних обмежень з будь-якою ціллю.

Важливою рисою SQLite є той факт, що в ній не використовується клієнт-серверний принцип. Мається на увазі, що рушій SQLite не працює з застосунком в ролі окремого процесу, а надає змогу працювати з бібліотекою, яка компілюється у складі програми. Після успішної компіляції рушій стає її складовою частиною. У такий спосіб в ролі протоколу для обміну даними можуть використовуватись функції, доступ до яких надає інтерфейс(API) бібліотеки SQLite. Таким чином зменшуються час відгуку застосунку, накладні витрати, а також стає можливим спрощення структури програми.

SQLite зберігає усі елементи БД (включно з таблицями, визначеннями, даними та індексами) в уніфікованому стандартному для неї файлі на тій машині, в серидовищі якої запускається додаток. Легкість та простота реалізації досягається завдяки блокуванню файлу, в якому зберігається БД перед стартом процесу виконання транзакції; принципи ACID-функцій досягаються в тому числі завдяки існуванню файлу-журналу.

Система надає можливість без жодних проблем зчитувати дані з однієї і тієї ж бази кільком процесам або потокам в один і той же момент. На відміну від процесу читання, процес запису в БД система надає змогу почати лише за умови, що в даний момент не обслуговуються інші запити, в іншому випадку спроба запису завершиться невдачею, після чого в програму повернеться код помилки процесу. Також в системі є можливим повторення запитів у разі помилки з певними інтервалами [13].

В коробці у системі є також виконуваний файл sqlite3, який є функціональною клієнтською частиною, яка надає можливість демонстрації реалізації функцій з основної бібліотеки. Клієнтська частина надає інтерфейс для роботи в командному рядку, і дозволяє звернення до файлу БД на основі типових функцій ОС.

Архітектура рушія надає можливість використовувати SQlite як на виділених машинах, що містять величезні масиви даних, так і на вбудовуваних (embedded) системах.

Особливості SQLite:

- транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення;
- встановлення без конфігурації – не потребує ані установки, ані адміністрування;
- реалізує значну частину стандарту SQL92;
- база даних зберігається в одному крос-платформовому файлі на диску;
- підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків;

– малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями;
– швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій;
– простий, легкий у використанні API;
– добре прокоментований сирцевий код зі 100% тестовий покриттям гілок;
– доступний як єдиний файл сирцевого коду на ANSI C, який можна легко вставити в інший проект;
– автономність: немає зовнішніх залежностей;
– поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

SQLite має репутацію вкрай надійної системи баз даних, яка знайшла широке застосування в багатьох уже звичних для людей електронних гаджетах і програмах, включно з деякими MP3-програвачами, iPhone, iPod Touch, Mozilla Firefox і інші [14].

2.2.3. Microsoft SQL Server

Microsoft SQL Server – розроблена та розповсюджувана корпорацією Microsoft комерційна система керування базами даних. Microsoft та Sybase спільно розробили та впровадили Transact-SQL - розширену версію SQL саме для даного продукту. Вона реалізована відповідно до ANSI / ISO стандартів, які застосовуються до структурованих мов запитів, де за основу взято SQL. Вона широко застосовується для створення баз даних різних масштабів: від невеликих БД для простих продуктів величезних сховищ даних, які за об'ємами вмісту досягають рівня підприємств та корпорацій. Впродовж довгого часу вдало витримує конкуренцію з іншими продуктами.

В якості мови, що буде використовуватись для запитів до бази даних Microsoft SQL Server застосовує розроблену саме для неї версію SQL. Вона була названа TRANSACT-SQL(скорочено T-SQL). T-SQL надає інтерфейс для ефективного

використання розширеного синтаксису процедур, що зберігаються і реалізує обмін даними БД з головним додатком через транзації. Робота Microsoft SQL Server та Sybase ASE з мережею відбувається з допомогою протоколу передачі даних з таблиць рівня додатку під назвою Tabular Data Stream (TDS).

Система надає можливість для віддзеркалення та кластеризації баз даних, де один кластер SQL є групою однаково налаштованих серверів. Усі вони об’єднані під єдиним для усіх віртуальним іменем. Дані в такій системі під час життєвого циклу розподіляються між серверами за IP-адресами машин кластеру. Задля стабільності та безпеки система у випадку відмови або збою роботи на одному з серверів групи система може автоматично перенести навантаження на інший сервер [15].

Таблиця 1

Порівняння систем керування базами даних Firebase Realtime Database, Microsoft SQL Server, SQLite

Назва	Firebase Realtime Database	Microsoft SQL Server	SQLite
Опис	Хмарна база даних реального часу.	Microsofts реляційна система керування базами даних	Широко використовувана система керування базами даних
Модель бази даних	Документне сховище	Реляційна система керування базами даних	Реляційна система керування базами даних
Хмарне сховище	Так	Ні	Ні
Схема даних	Вільна схема	Потрібна	Потрібна
SQL	Не використовується	Використовується	Використовується
Зовнішні ключі	Ні	Так	Так

Продовження таблиці 1

Концепція транзакцій	Так	ACID	ACID
Паралелізм	Так	Так	Так

2.3. Вибір картографічного сервісу

2.3.1. Google Maps

Google Maps – комплексна система сервісів, технологій, засобів аналітики та додатків від Google, які були розроблені на основі картографічного сервісу компанії. Система включає в себе мапу Землі, Місяця та Марса, а також їх супутникові знімки, які повністю покривають територію нашої планети. Користувачі мають змогу використовувати додаткові інтегровані в додаток з мапами сервіси, як от бізнес-довідник, карта автомобільних та залізничних доріг, тощо.

Однією з функцій є «Громадський Транспорт», запущений компанією як експеримент у 2005 році. Повноцінною частиною Google Maps сервіс став у 2007 році. Google активно кооперується величезною кількістю транспортних відомств з усього світу, а також збирає величезну кількість даних про переміщення та маршрути. Це дозволяє їй надавати користувачам актуальну так корисну інформацію про графік, маршрут, напрям та частоту руху громадського транспорту.

Інформацію можна отримати в усіх браузерах, які можуть відкрити сторінку Google Maps, а також у додатках для мобільних телефонів на операційних системах Android та iOS.

Google Street View – частина комплексу Google Maps, яка реалізує можливість відображення та переглядання вулиць від першого обличчя на екрані користувача. Щоб створити цей сервіс, кожну вулицю, побачити яку користувач має змогу в програмі, особливим обладнанням в режимі реального часу знімали методом кругового фотографування. В результаті цього для кожної точки, яка доступна для перегляду в

сервісі створювалась власна сферична панорама. З допомогою великої кількості таких панорам, прив’язаних до географічних координат, створюється ефект переміщення по вулицях чи місцевості. Користувач має змогу перемикатись між панорамами, для чого Google надає простий та зрозумілий інтерфейс.

Google надає потужності свого сервісу до послуг розробникам програмного забезпечення за допомогою Google Maps Api. Це інтерфейс для створення, додавання, редагування та кастомізації мап під потреби розробників, який надає широкий вибір інструментів для взаємодії. Компанія надає досить великі квоти для безоплатного використання своїх сервісів, що робить їх ідеальним вибором для навчання та розробки тестових додатків. Через інтерфейс розробник має змогу використати інструменти для побудови маршрутів, відображення та редагування маркерів, зміни зовнішнього вигляду та переміщення мапи, фільтрування об’єктів за їх місцезнаходженням, обчислення відстаней тощо.

Основними перевагами даного сервісу є:

– широкий вибір доступних для використання функцій для роботи з мапою;
– можливість зручного редагування та додавання елементів на мапу;
– зібрана та своєчасно оновлювана документація.

2.3.2. Open Street Map

OpenStreetMap – міжнародний, безкоштовний, не комерційний та відкритий до покращення, співпраці та розробки картографічний сервіс. Він не є мапою в прямому сенсі цього слова, а зібраним та обробленим набором геопросторових даних. Так як проект не є комерційним, збором, аналізом та введенням даних до мапи займається спільнота волонтерів, які на добровільних засадах займаються покращенням сервісу.

Проект дозволяє використання усіх своїх даних, так як розповсюджується на умовах ліцензії Open Database License. Це дає змогу використовувати дані цього проекту

з будь-якою ціллю, в тому числі і для комерційної діяльності за умови зазначення їх джерела.

База геопросторових даних сервісу включає в себе інформацію про окремі точки, а також характеристики об'єктів більш високого порядку – лінії, які прокладаються між точками та з'єднують їх. Іншим типом об'єктів є зв'язки, які можуть складатись з точок та ліній, а також містити інформацію про усі атрибути зазначених об'єктів.

Основним джерелом наповнення проекту геопросторовими даними є зібрані волонтерами треки. Їх збирають за допомогою пристроїв, що обладнані GPS-приймачами, після чого ці дані завантажуються до бази сервісу. В подальшому їх використовують для прив'язки доріг на мапі та для перевірки того, чи правильно були прив'язані супутникові знімки до місцевості. Часто ці треки є єдиним джерелом інформації про місцевість, у тому випадку, коли супутникові знімки, що прив'язані до неї, мають погану якість, або ж взагалі відсутні.

Інтерфейс сервісу створений за допомогою відкритої бібліотеки Leaflet.js, яка дозволяє створення кастомізованих інтерактивних мап.

2.3.3. Яндекс Карти

Яндекс.Карти – сервіс від компанії Яндекс, що надає пошуково-інформаційні можливості для користувачів. Був відкритий у 2004 році.

Веб-ресурс дозволяє отримати інформацію про дороги, об'єкти інфраструктури, дорожній трафік, а також дозволяє переглядати панорамні знімки. Для країн східної Європи, частини країн близького та середнього сходу і суміжних з ними сервіс використовує власні мапи, для інших країн до 2017 року мапи постачала компанія НАВТЕК, після чого Яндекс перейшов на власні дані.

Карти доступні у чотирьох варіаціях:

– народна карта – наповнюється даними власне користувачами;
– схеми – стандартний вигляд мапи;
– супутниковий знімок;
– гібрид схем і супутникового знімку.

Набір інструментів для роботи з картами надзвичайно широкий, але в повному обсязі доступний у певній кількості найпопулярніших міст. Серед доступних можливостей перегляд зображень з веб-камер, індикатор заторів, дані з якого можуть враховуватись при побудові маршруту та обчисленні оптимального шляху.

Компанія має широкий спектр додатків для смартфонів, які надають доступ до різноманітних гео-сервісів: Яндекс.Карти, Яндекс.Пробки, Яндекс.Транспорт, Яндекс.Місто.

2.4. Середовище розроблення

Для розроблення продукту мною був вибраний продукт компанії JetBrains – WebStorm. Це середовище було розроблене для створення програмних продуктів на мові JavaScript та похідних від неї фреймворках та бібліотеках. Програмний комплекс надає широкий вибір інструментів для програмування не лише клієнтської сторони коду, яка створюється з допомогою JavaScript, а й створення серверних додатків на back-end фреймворку Node.js, мобільних додатків з допомогою технології React Native.

Середовище містить велику кількість вбудованих плагінів для роботи з веб-технологіями, зокрема мовою розмітки веб-сторінок HTML та мови стилів CSS. Розумний редактор дозволяє підключати і потрібні користувачу плагіни для роботи з практично будь-яким JavaScript фреймворком чи бібліотекою, а також для бібліотек, що розширюють можливості CSS: Sass, Stylus, Less.

WebStorm також дозволяє налаштувати автоматизовану компіляцію таких інструментів, як Sass та TypeScript, що робить процес програмування простішим та зручнішим для розробника.

Варто зазначити, що безкоштовно користуватись продукцією компанії JetBrains можна лише з обмеженим набором інструментів, але для студентів компанія надає можливість оформлення студентської підписки на продукти. Це дозволяє використовувати увесь комплекс професійних інструментів для розробки.

WebStorm, як і інші продукти компанії JetBrains, був розроблений на платформі IntelliJ IDEA. Серед стандартних можливостей даного середовища варто виділити такі:

– автодоповнення коду – звична риса більшості середовищ розробки, але завдяки інтеграції з більшістю фреймворків та бібліотек, а також гнучкому підключенні плагінів, при автодоповненні WebStorm самостійно враховує логіку підключених бібліотек та модулів, що робить його особливо зручним для розробки на JavaScript та похідних технологіях;
– перевірка на помилки на підсвітка коду – може налаштовуватись користувачем і враховувати особливості технологій, з допомогою яких розробляється програма;
– зручна система рефакторингу, яка враховує підключені до середовища засоби розробки і модулі, і дозволяє безпечно та з урахуванням усіх факторів міняти код.

2.5. Додаткові технології та бібліотеки

2.5.1. Firebase Authentication

Для повноцінної реалізації розроблюваного проекту необхідна система реєстрації, авторизації та аутентифікації.

З цією метою було вирішено скористатись одним з інструментів Firebase Console – Firebase Authentication. Він дозволяє повністю перекласти процеси авторизації, реєстрації та аутентифікації з сервера на власні потужності, і надає змогу користуватись своїм інтерфейсом без прокладки на стороні сервера прямо з клієнтського коду. При потребі розробник може скористатись простим у встановленні на налаштуванні SDK як на серверній стороні, так і на клієнті.

Важливою перевагою даної системи є зручний спосіб авторизації через більшість популярних соціальних мереж та платформ, за допомогою електронної пошти, номеру телефону та паролю. Це дозволяє значно прискорити процес створення продукту на зосередитись на інших задачах, замість написання коду для процесів авторизації.

Firebase Authentication можна легко інтегрувати у інші сервіси Firebase. Внаслідок цього розробник отримує можливість працювати з даними користувачів не лише засобами Firebase Authentication, а й керувати ними як записами в базах даних, тощо. Сервіс використовує стандарти OAuth 2.0 і OpenID Connect, завдяки чому легко інтегрується з серверною частиною додатку.

2.5.2. Firebase Analytics

Firebase Analytics дає змогу розробнику зрозуміти, як користувачі використовують розроблений додаток. Основі дані доступні прямо з Firebase Console, що особливо зручно, якщо розробник використовує Firebase в тому числі для хостингу. Також для різноманітних потреб доступні SDK для мобільних додатків та серверних застосунків, які дозволяють гнучке налаштування фіксації наперед визначених розробником чи системою подій і властивостей користувача. Розробник має змогу створювати свої власні події для моніторингу важливих у розроблюваному додатку показників.

Дані доступні у зручному вигляді в будь-який час у Firebase Console. На ній відображаються усі зібрані дані – від зведених, таких як кількість активних користувачів та демографічна статистика, до більш докладних даних, наприклад, кількість перегляду тої чи іншої сторінки додатку.

Аналітика зручно інтегрується з більшістю інших сервісів, які доступні у Firebase Console. Вона допомагає зрозуміти та оцінити поведінку користувачів, що дозволяє оперативно реагувати на потреби у зміні чи покращенні певних функцій додатку.

Важливою задачею розроблюваного додатку є збір та аналіз даних про найбільш цікавий користувачам контент, а також дослідження території з найбільшою кількістю зацікавлених користувачів, а Firebase Analytics дозволить спростити цей процес.

Оскільки основною з задач розроблюваного додатку є можливість якісно аналізувати дані про велосипедистів міста, використання Firebase Analytics дозволяє значно полегшити цю задачу та надає великий вибір інструментів.

2.6. Висновки

У даному розділі я розглянув основні інструменти та технології для створення веб-застосунків. Вихідні дані були систематизовані та проаналізовані з метою подальшого використання у роботі визначеного набору інструментів, які будуть використовуватись для розробки онлайн сервісу агрегатора подій.

Основною мовою розробки було вирішено обрати JavaScript, основними аргументами для цього рішення були:

- часте використання даної мови у веб-програмуванні та її пристосованість саме до цієї області розробки;
- високий рівень розвитку мови та величезна спільнота розробників;
- велика кількість побудованих на цій мові різноманітних бібліотек та фреймворків, які можна вільно використовувати;
- обрана для проекту база даних Firebase має зручний для інтеграції з веб-технологіями інтерфейс.

В якості основного фреймворку для розроблення було обрано Angular 8, так як:

- даний фреймворк є одним з найпопулярніших для розробки на JavaScript;
- якісно та повноцінно оформлена документація;
- висока кількість готових рішень “з коробки”;
- простота в оновленні та підтримці довгострокових проектів;

– наявність повноцінних офіційних бібліотек для зручної взаємодії з обраною для проекту базою даних Firebase.

Для використання в якості бази даних було вирішено обрати Firebase, у зв’язку з тим, що:

– дозволяє оновлювати дані для усіх користувачів в режимі реального часу;
– має зручну інтеграцію з усіма сервісами Google, а також надає велику кількість інструментів для керування даними веб-застосунків;
– має зручний інтерфейс для взаємодії з Google Cloud для збереження статичних даних користувачів;
– надає інструменти для налаштування процесів аутентифікації, авторизації та реєстрації в додатку, а також їх зручне оновлення з інформацією в базі даних.

Картографічним сервісом було обрано Google Maps, оскільки:

– він має найбільше покриття земного шару серед усіх аналогів;
– широкі можливості для кастомізації мапи у власноруч створених додатках;
– офіційні, повноцінні та зручні для використання бібліотеки для обраних мною в проекті технологій.

3. ОПИС РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ

3.1. Загальний опис веб-додатку

Дану систему є веб-ресурсом, який було розроблено з широким набором функціональних можливостей та з урахуванням сучасних стандартів користувацьких інтерфейсів.

Неавторизований користувач має доступ лише до базових функцій додатку, для використання повного набору інструментів, таких як пошук, фільтрація, додавання та редагування подій на веб-ресурсі йому необхідно увійти в систему або зареєструватись. При початковому завантаженні користувач бачить лише мапу та базові події.

У системі існує два типи користувачів: звичайний користувач та модератор. Після реєстрації кожен користувач отримує змогу:

- переглядати події на мапі;
- виконувати пошук подій за назвою;
- виконувати пошук подій за датою проведення;
- виконувати пошук подій у певному місті;
- додавати події до списку улюблених.

Особливою можливістю для кожного зареєстрованого користувача є створення власних подій. Цю можливість було створено для тих користувачів, які бажають розміщувати інформацію про власні події, або ж для представників таких осіб. Для цього при створенні події заповнюється також інформація про тематику діяльності організатора і зазначаються його контактні дані. Після перевірки доданої події одним із Адміністраторів сайту і її ухвалення вона відобразиться на мапі.

Для взаємодії з подіями користувачам надані такі можливості:

- створювати власні події;
- відобразити подію на мапі;
- редагувати дані про подію;

– виділяти маркер події на мапі;

– додавати подію в список улюблених.

3.2. Функціональні вимоги до веб-ресурсу

Неавторизовані користувачі мають можливість:

– реєстрації;

– авторизації;

– перегляду подій на мапі;

– перегляду інформації про подію.

Авторизовані користувачі мають можливість:

– пошуку подій на мапі за назвою;

– пошук подій на мапі за типом події;

– додавання подій до списку улюблених;

– перегляду списку улюблених подій;

– редагування профілю;

– отримання сповіщень про нові події;

– інформування про порушення подією правил сайту;

– пошуку подій на мапі за датою проведення;

– пошуку подій на мапі за місцем проведення.

– додавання власних подій на мапу;

– додавання інформації про місце, час проведення, ціну події, тип події, зображення події;

– надсилання сповіщень про нові події користувачам;

– редагування даних про подію;

– додавання посилання на квитки чи офіційну сторінку події.

Авторизовані Адміністратори мають право:

– підтверджувати створення події користувачем;

– розглядати скарги на події;

– видаляти події та профілі користувачів, що не відповідають правилам веб-ресурсу.

3.3. Архітектура веб-додатку

Для розроблення продукту було обрано клієнт-серверну архітектуру.

Клієнт-серверна архітектура – це такий спосіб організації структури веб-додатків в якій завдання, поставлені перед системою умовно діляться між двома підсистемами: клієнтом та сервером.

До сервера звертаються безліч клієнтів та часто він зберігає дані, тож він є “центральною” частиною. З цього слідує, що сервер може існувати без клієнтів, а от клієнти без сервера не можуть. Клієнт – програма, примірників якої зазвичай "більше", ніж серверів. Наприклад, до одного веб-сервера (що зберігає сайт) може підключатися велике число веб-браузерів. Зазвичай саме з програмою-клієнтом працюють користувачі (тому її так і називають), з іншого боку клієнтом сервера може бути і робот.

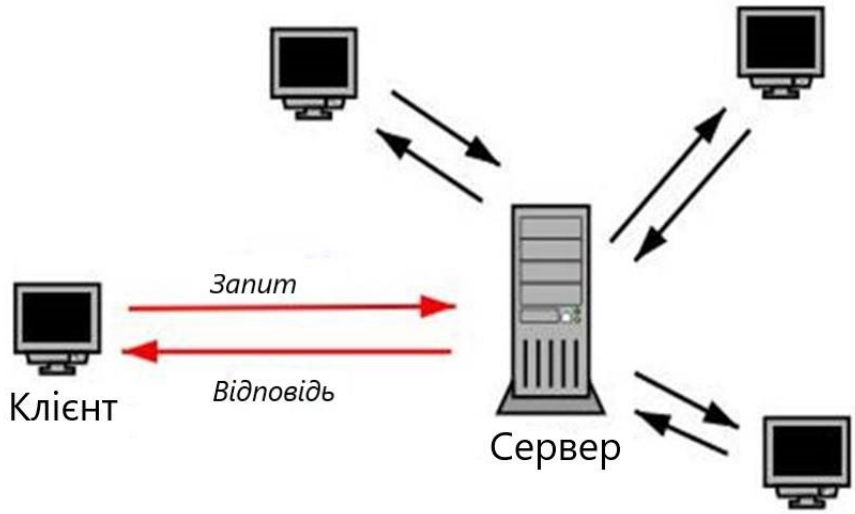


Рис. 5. Клієнт-серверна архітектура

Під час розроблення клієнтської частини веб-ресурсу було використано шаблон проектування MVC (Model-View-Controller). Для його реалізації застосовано фреймворк

Angular 8.

Структура проекту була продиктована загальноприйнятими та рекомендованими розробниками фреймворку практиками розроблення додатків на фреймворку Angular 8, а також необхідністю розділення функціональності системи та відображення інформації про сутності платформи для кожного окремого користувача. Загалом платформа розділена на наступні логічні частини:

1. Компонент – основна структурна одиниця додатку, що написаний на Angular 2+. Являє собою суміш контролера та шаблону. Він включає в себе елементи відображення, стилізації та контролер для обробки даних та їх відображення в шаблоні. Для кожного компоненту виділено окрему папку, в якій знаходяться HTML-файли, файли стилів, файл контролера та тестування. Основними засоби налаштування компоненту є селектор, шаблон, входи і виходи даних, директиви та інші компоненти, що можуть бути використані в батьківському компоненті.
2. Модуль – складається з одного або декількох компонентів, містить власну логіку роутингу (вирахування сторінок, які пройдуть процес рендерингу для відображення в браузері), визначає порядок використання компонентів, що були декларовані всередині нього та їх взаємодію з іншими модулями та компонентами. Може містити також директиви, сервіси, пайпи. Кореневий модуль включає в себе налаштування та створення додатку Angular 2+, підключення усіх модулів у ієрархічній системі.

Елементи інтерфейсу розділено на чотири основні групи:

1. Шаблони – є статичними частинами клієнтської частини, що відповідають за відображення окремих частин системи, функціонально виконують задачі рендерингу, підписки на події, використовують значення даних, які їм надає контролер, а також відображає зміну даних внаслідок роботи контролера.

2. Компоненти – являються сумішшю шаблону та контролеру, є найнижчою ланкою ієрархії роутингу, мають власний життєвий цикл, можуть використовувати сервіси для отримання та обробки даних а також містити власну логіку.
3. Сервіси – спектр класів, які надають можливість виконувати обробку, отримання та відправку даних, а також відокремити необхідні операції від логіки власне компонента, тобто інкапсулювати їх. Їх використання в різних компонентах можна налаштувати з допомогою модулів.
4. Директиви – описують зміни поведінки або перетворення DOM-дерева, пов'язані з користувацьким атрибутом, ім'ям елемента, або css-класом. Директиви дозволяють розширити HTML-синтаксис в декларативної формі. Angular поставляється з широким набором директив, корисних для створення веб-додатків, а також можливістю створення власних.
5. Контролери являються файлами TypeScript, кожен з яких є частиною певного компоненту та відповідає за обробку та відображення в шаблоні, а також контроль над його життєвим циклом. Надає можливість підключати до компоненту сервіси.

Модулі слугують найвищою ланкою в побудові ієрархії для взаємодії компонентів, сервісів та директив, і можуть включати в себе сервіси та компоненти з інших модулів. Експорт компонентів налаштовується також в модулях.

Схема взаємодії модулів, компонентів на сервісів представлена на рис. 6.

Загальну структуру проекту клієнтської частини веб-ресурсу «Мапа подій» наведено на рис. 7.

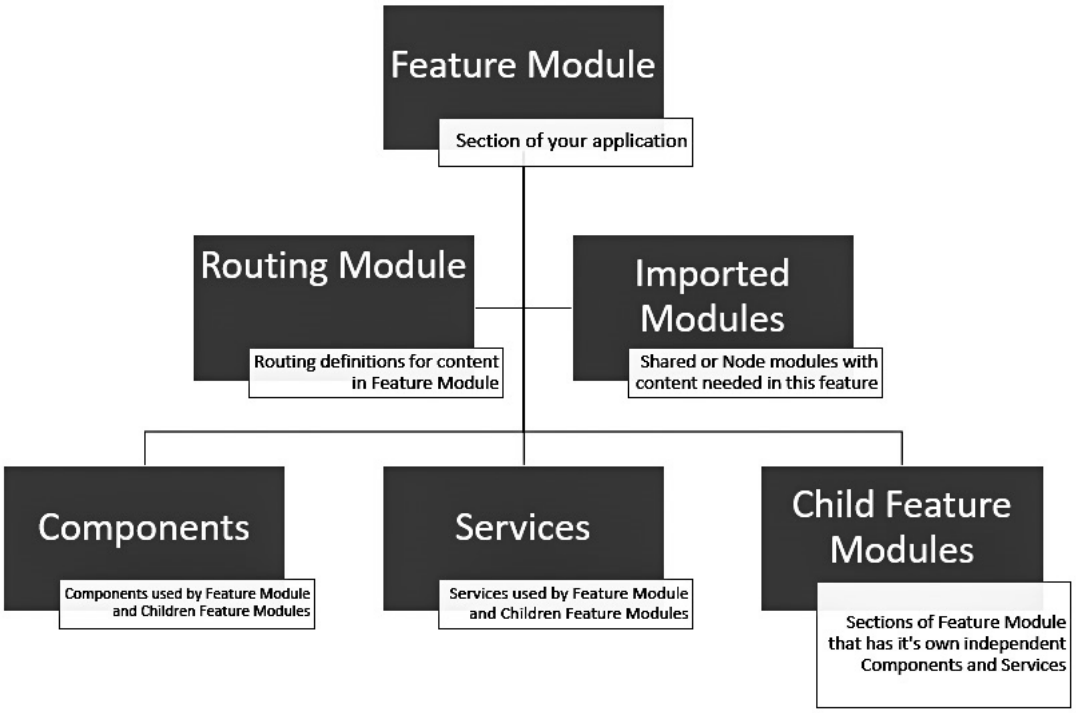


Рис. 6. Зв’язки між складовими Angular 2+ додатку

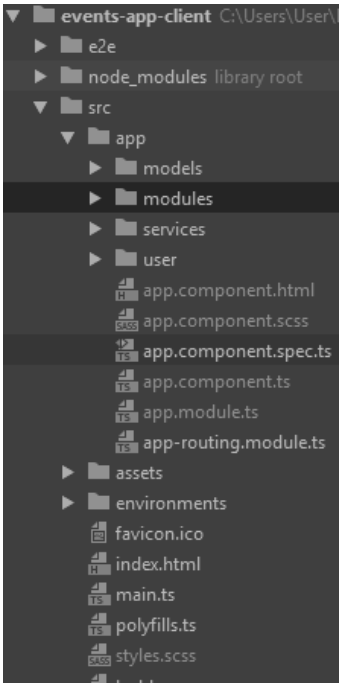


Рис. 7. Загальна структура клієнтської частини додатку

3.4. Опис модулів програмного забезпечення

У структурі веб-сервісу можна чітко виділити такі важливі розроблені модулі:

- модуль авторизації та реєстрації;
- модуль геокодування;
- модуль компонентів відображення;
- модуль зв’язку з БД;
- модуль серверного рендерингу.

Далі буде розглянуто найважливіші модулі розробленого застосунку та їх взаємодію.

3.4.1. Модуль авторизації та реєстрації

Сутність користувача складається з таких полів:

- display_name – ім’я користувача, що відображається на сайті;
- email – адреса електронної адреси користувача;
- photoURL – фотографія, яка відображається на профілі користувача;
- uid – унікальний ідентифікатор користувача;
- favorites – список улюблених подій користувача;
- events – список подій організатора;
- admin – булеве значення, індикатор прав адміністратора;
- organizer – булеве значення, означає наявність власних подій.

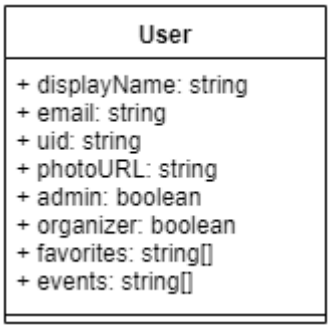


Рис. 8. Сутність користувача

При реєстрації користувач може скористатись опціями авторизації з допомогою акаунта популярних соціальних мереж та інших сервісів: Facebook, Twitter, Github, Microsoft. Після реєстрації з клієнта на сервер відправляються дані та зберігаються з допомого сервісів firebase authentication, а також дані користувачів дублюються в БД.

3.4.2. Модуль серверного рендерингу

При застосуванні технологію SSR(Server Side Rendering) сервер після запиту до нього генерує усю HTML-сторінку. Це виключає необхідність додаткових запитів даних з боку клієнта, так як сервер бере всю роботу на себе, перш ніж відправити відповідь. Такий підхід дозволяє домогтися швидкого рендерингу першої сторінки і рендерингу змісту першої сторінки. Швидке відображення першої сторінки може бути критично важливим для участі користувача, особливо в тому випадку, коли з певних причин сторінка на стороні користувача завантажується та оброблюється повільно. Таке трапляється при побудові складних сучасних веб-ресурсів, де велика частина логіки оброблюється в браузері. Приблизно 53% користувачів переривають з'єднання та виходять з сайту, якщо відповідь прийшлось очікувати більше трьох секунд.

Ця технологія прискорює отримання користувачем сторінки, яку він хоче переглянути.

У розробці використовувався Universal – модуль Angular для використання SSR.

Існує три основні причини створення Universal версії програми:

– сприяти веб-сканерам через пошукову оптимізацію (SEO);
– покращити продуктивність на мобільних та малопотужних пристроях;
– якнайшвидше показувати першу сторінку за допомогою першого вмісту (FCP).

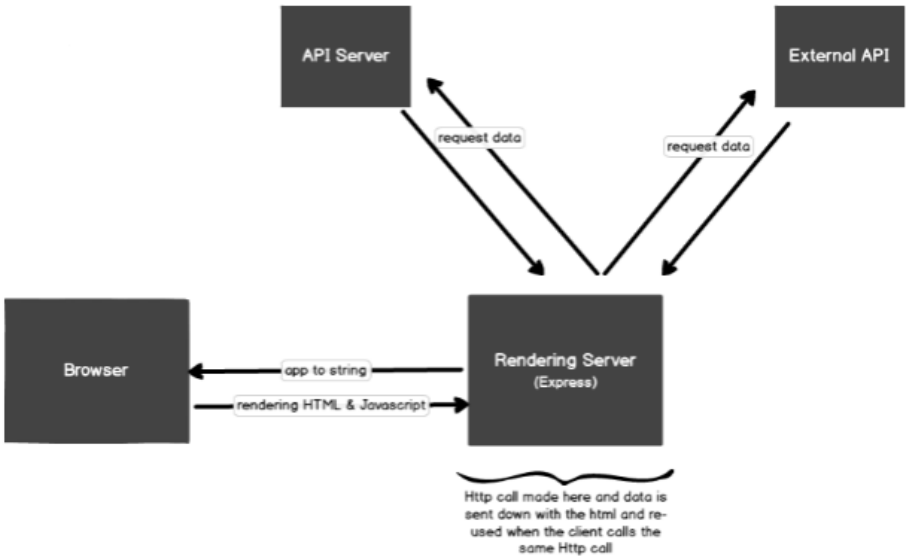


Рис. 9. Angular Universal

3.4.3. Модуль геокодування

Функції геокодування у додатку виконують firebase сервіси від Google: Google Maps та Google Geocoder. Основна їх функція полягає у визначенні координат за назвою місця, введеного користувачем. Точні координати необхідні для виставлення маркерів у потрібних місцях на мапі за допомогою Google Maps api. Також Google Geocoder дозволяє вираховувати відстань між координатами, що дозволяє виконувати пошук координат, що входять в коло певного радіусу пошуку.

3.5. Висновок

В даному розділі проведено загальний опис розробленої клієнтської частини веб-ресурсу “Мапа подій”, визначено функціональні вимоги, які повинен задовольняти

реалізований додаток для неавторизованих користувачів та двох типів зареєстрованих акаунтів: Адміністратора на звичайного користувача. Дані вимоги повністю реалізовано у розробленому веб-застосунку.

Також розглянуто і описано загальну структуру програмного проекту, а саме Model-View-Controller, що наразі є стандартом для веб-розробки, оскільки забезпечує необхідний рівень інкапсуляції, мінімізацію ймовірності помилок та читабельність коду для подальшої розробки.

Також описано загальні риси використаної в проекті клієнт-серверної архітектури.

Для реалізації даного шаблону було використано фреймворк Angular8 та мову JavaScript і її надбудову TypeScript.

Для стилізації веб-сторінок підключено зовнішню бібліотеку Angular Material, що надає зручні класи, компоненти та директиви для стилізації додатку.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ

Систему розроблено за стандартами шаблону MVC. Робочий потік застосунку було побудовано згідно з сучасними стандартами розроблення інтерфейсів веб-додатків. Загальна схема типової взаємодії користувача із веб-ресурсом складається з наступних кроків:

1. Вхід на сторінку веб-застосунку.
2. Завантаження основних модулів додатку.
3. Контролер за допомогою сервісу надсилає запит на серверну частину для отримання даних з БД.
4. Відображення на мапі маркерів згідно з отриманими даними та їх перегляд користувачем.

Після цього у користувача є можливість авторизації, яка розширює доступні можливості для роботи з веб-ресурсом.

4.1. Робочий потік веб-додатку

На головній сторінці міститься головна мапа, яка відображає події, а також основні кнопки логіну та реєстрації (рис. 10). Після натискання на них користувач переходить до зручної для нього форми авторизації: через одну з соціальних мереж чи включених веб-сервісів, або ж за допомогою пошти. Форма для реєстрації включає в себе:

– поле вводу для електронної пошти;
– замасковане поле вводу для паролю;
– кнопка підтвердження форми;

Після входу в додаток користувач залишається на тій же сторінці, але з розширеними функціональними можливостями. Для звичайних зареєстрованих користувачів до них входить фільтрація, пошук подій та додавання подій до списку

улюблених. Улюблені події, а також засоби для пошуку та фільтрації знаходяться в боковому меню.

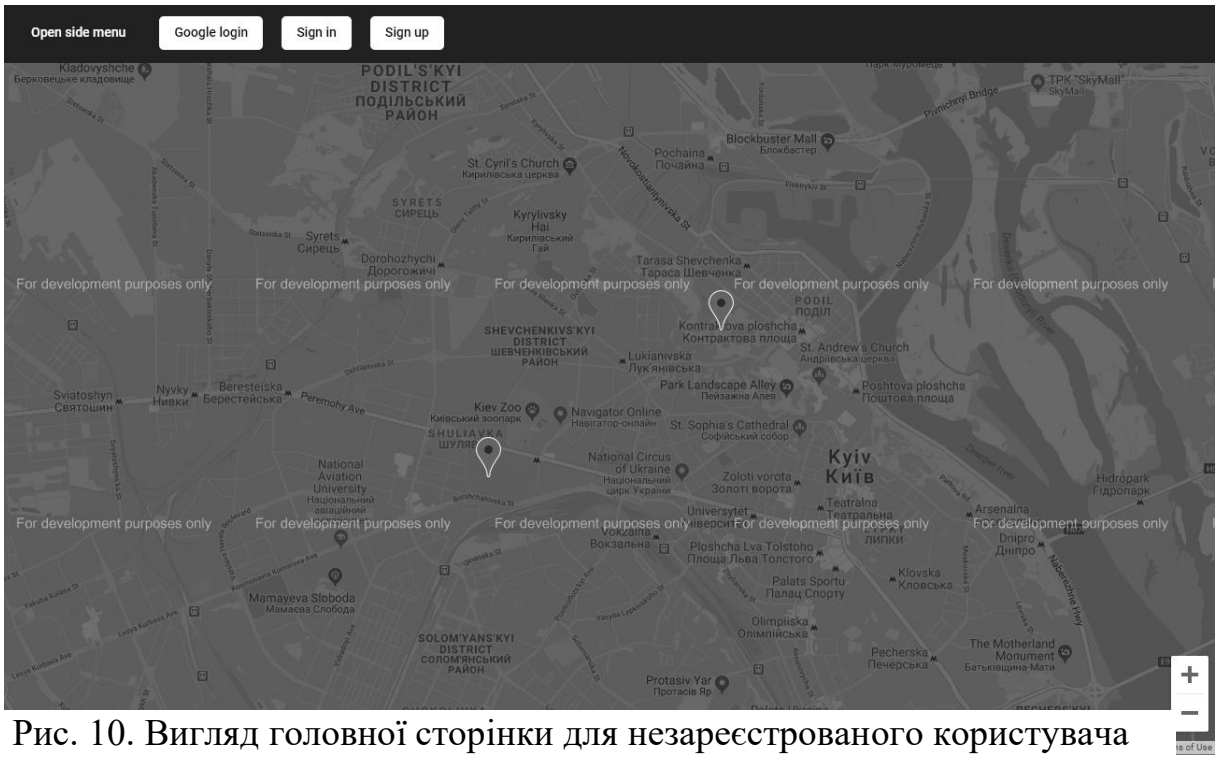


Рис. 10. Вигляд головної сторінки для незареєстрованого користувача

Користувач має можливість натиснути на маркер будь-якої події на карті та отримати розгорнуту інформацію про неї (рис. 11):

1. Повна назва події.
2. Ім’я організатора події чи організації, яка її створила.
3. Тип події: кінопоказ, вечірка, концерт.
4. Дата проведення.
5. Інформація про тип оплати події: безкоштовна, вхід за символічну пожертву, платна.
6. Інформація про вартість події.
7. Фото події.
8. Опис події.
9. Посилання на офіційну сторінку чи платформу для продажу квитків.

Після авторизації у користувача з’являється можливість створити власну подію,

для чого йому потрібно надати інформацію про свої контакти, напрям діяльності та назву організації, якщо така наявна.

Рис. 11. Розгорнуте вікно з інформацією про подію

При створенні події організатор має змогу заповнити та надати для перегляду такі дані:

- назва події;
- тип події – надає можливість для пошуку за тегом;
- дата та час проведення;
- безкоштовність події;
- вартість події;
- опис події;
- фото події;

– посилання на офіційну сторінку чи платформу для продажу квитків.

В формі доступний селектор дати да тегів для події (рис. 12).

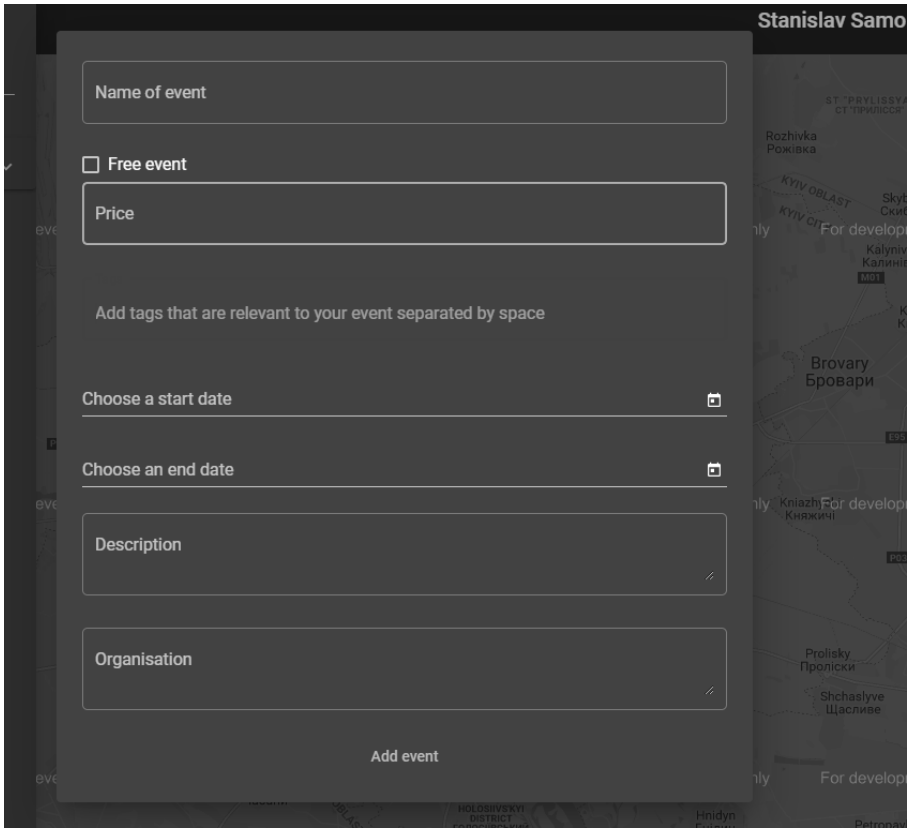


Рис. 12. Форма додавання власної події з полями

4.2. Тестування веб-додатку

Контроль за якістю є постійною та важливою частиною життєвого циклу розроблення та підтримки програмного забезпечення та входить в кожен етап створення веб-додатку, як завершальна частина перед затвердженням імплементації функціональності.

Тестування направлено на виявлення помилок наступних типів:

– інтерфейсних (помилки стилізації, відображення сторінок);
– функціональних (перевірки правильності роботи частин додатку та інформаційних потоків);

– авторизаційних (захищеність від доступу до окремих елементів інтерфейсу та конфіденційних даних для незареєстрованих або з недостатнім рівнем доступу користувачів).

Одним із способів тестування ПЗ було ручне тестування, за допомогою якого вводилися різні дані до текстових полів, зокрема некоректні дані, пусті значення тощо. За результатами цього тестування було додано повідомлення для користувачів про відповідні помилки, які виникають при введенні ними некоректних даних.

4.3. Порівняння розробки з існуючими аналогами

Розроблений веб-додаток відповідає усім визначеним вимогам, а саме:

– можливість переглядати події в усьому світі на інтерактивній мапі;
– можливість пошуку та фільтрації подій за різними параметрами;
– можливість додавання власних подій;
– можливість додавання подій до списку улюблених.

Окрім цього було усунуто наступні недоліки альтернативних програмних засобів:

– вузька направленість аналогів;
– відображення контенту лише для певної території;
– складність створення та просування власних подій;
– велика кількість реклами;
– складність пошуку та фільтрації контенту;
– складний та не інтуїтивний інтерфейс.

Провівши аналіз розробленого програмного засобу, можна стверджувати, що він є достойним конкурентом для інших додатків, які є онлайн-ресурсами для планування дозвілля.

4.4. Рекомендації щодо подальшого вдосконалення

Розроблений веб-застосунок можна застосовувати для просування власних подій

локальними організаторами, оскільки додаток надає достатньо функціональних можливостей для створення та керування подіями.

Отже, розроблене програмне забезпечення можна і потрібно вдосконалювати, розширюючи його функціональність для подальшого використання для просування подій локальних організаторів та великих комерційних компаній, що прагнуть зробити популярнішими свої події та послуги. Для цього можна надати наступні можливості:

- додавання віртуального рахунку організаторам;
- додавання можливості просування власних подій;
- виділення події на мапі за додаткову плату;
- інтеграція сервісу як частини популярних соціальних мереж та сервісів для продажу квитків;
- розширення аналізу аудиторії сайту;
- реалізація багатомовного інтерфейсу;
- додання системи сповіщень електронною поштою для користувачів на різноманітні події;
- додавання системи підписки на профіль організаторів;
- прив’язування банківських карток до сервісу для організаторів;
- розробка альтернативних версій додатку для різноманітних мобільних платформ.

ВИСНОВКИ

Метою дипломного проекту була розробка веб-ресурсу «Мапа подій», яка надавала б можливість користувачам переглядати події в усьому світі, виконувати операції пошуку та фільтрації за заданими параметрами.

Після проведення аналізу існуючих програмних аналогів, було зроблено висновок, що система має бути розроблена у вигляді веб-застосунку, що повинен мати зручний та зрозумілий інтерфейс, який мінімізуватиме помилки користувачів, та надаватиме надійну широку варіативність дій користувачам та організаторам.

За результатами проведеного аналізу засобів реалізації, а саме мов програмування, бібліотек та фреймворків було прийнято рішення розробити клієнтську частину веб-ресурсу «Мапа подій» мовою JavaScript за допомогою фреймворку Angular 8 за архітектурою шаблону MVC.

Розроблений додаток надає:

– можливість реєструватися і входити на сайт;

– можливість переглядати, шукати та фільтрувати події за заданими параметрами;

– можливість додавати власні події;

– користувацький інтерфейс, спроектований за сучасними стандартами UI та UX.

Розробку програмного забезпечення виконано у повному обсязі та у відповідності до сформованих вимог. Тестування веб-застосунку виконано за затвердженою програмою та методикою тестування.

Використання розробленого веб-додатку забезпечить користувачам комфортний та зручний сервіс для планування дозвілля, що є простим у керуванні та доступним у різних куточках світу.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

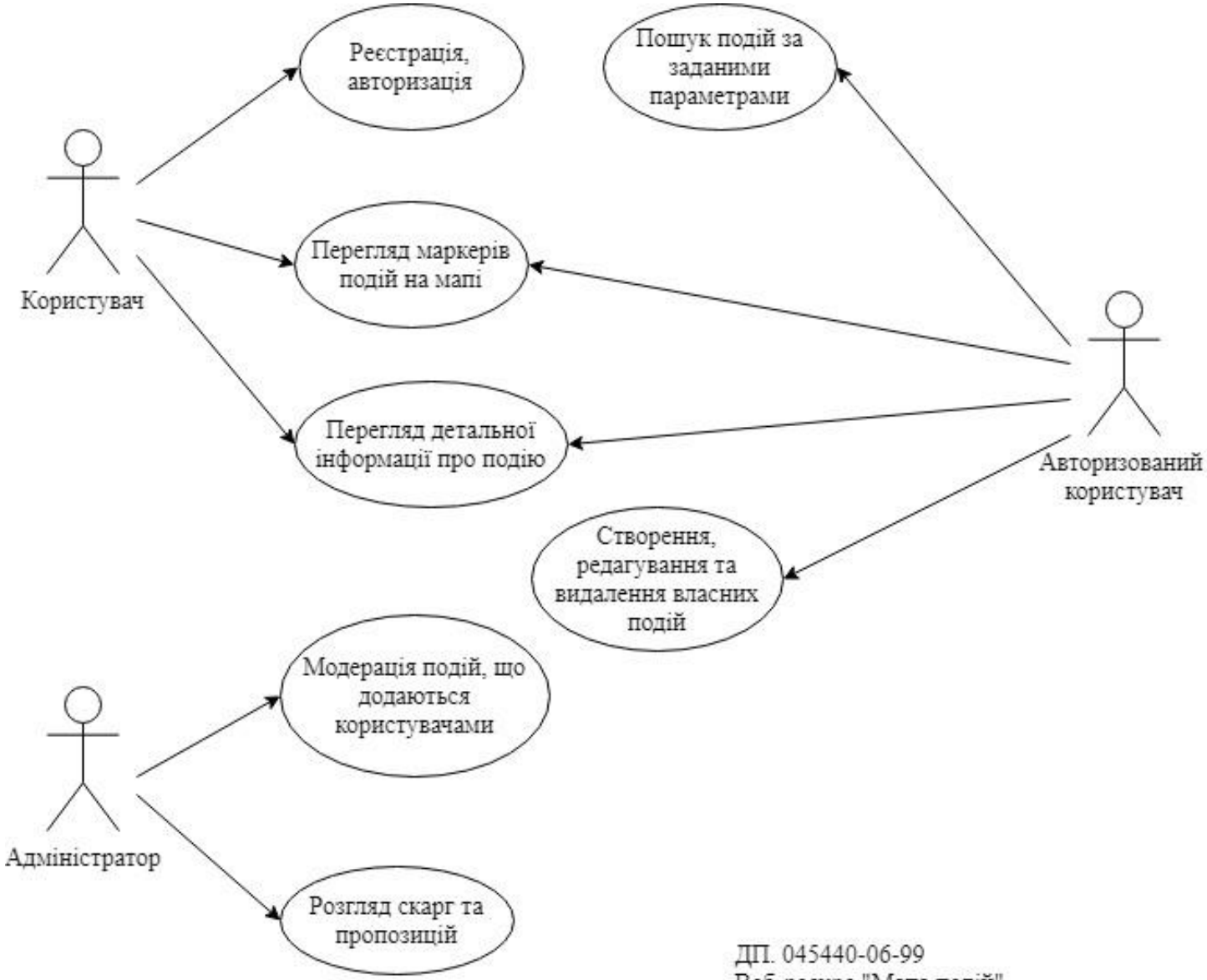
1. Google map help. View traffic, satellite, terrain, biking, and transit [Електронний ресурс]. — Режим доступу: https://goo.gl/9zxHhZ. Дата доступу: травень 2017. Назва з екрану.

2. Firebase [Електронний ресурс]. — Режим доступу: https://goo.gl/nJXHD3. Дата доступу: травень 2017. Назва з екрану.
3. SQLite [Електронний ресурс]. — Режим доступу: https://goo.gl/cezinW. Дата доступу: травень 2017. Назва з екрану.
4. SQL Features That SQLite Does Not Implement [Електронний ресурс]. — Режим доступу: https://goo.gl/vpNWZa. Дата доступу: травень 2017. Назва з екрану.
5. Microsoft SQL Server [Електронний ресурс]. — Режим доступу: https://goo.gl/v7bS8g. Дата доступу: травень 2017. Назва з екрану.
6. Работа с Google Maps API [Електронний ресурс]. — Режим доступу: https://goo.gl/2vYMhN. Дата доступу: травень 2017. Назва з екрану.
7. OpenStreetMap [Електронний ресурс]. — Режим доступу: https://goo.gl/31LJNd. Дата доступу: травень 2017. Назва з екрану.
8. Яндекс.Карти [Електронний ресурс]. — Режим доступу: https://goo.gl/M2z3ig. Дата доступу: травень 2017. Назва з екрану.
9. Firebase Authentication [Електронний ресурс]. — Режим доступу: https://goo.gl/pFbjcA. Дата доступу: травень 2017. Назва з екрану.
10. Google Analytics for Firebase [Електронний ресурс]. — Режим доступу: https://goo.gl/wQvJGG. Дата доступу: травень 2017. Назва з екрану.
11. Migrating to Standart SQL [Електронний ресурс]. — Режим доступу: https://goo.gl/DUuSpn. Дата доступу: травень 2017. Назва з екрану.
12. Google-json [Електронний ресурс]. — Режим доступу: https://goo.gl/KlQ3rm. Дата доступу: травень 2017. Назва з екрану.
13. Cloud database [Електронний ресурс]. — Режим доступу: https://goo.gl/4nBIuy. Дата доступу: травень 2017. Назва з екрану.
14. Firebase Cloud Messaging [Електронний ресурс]. — Режим доступу: https://goo.gl/SaL8v4. Дата доступу: травень 2017. Назва з екрану.

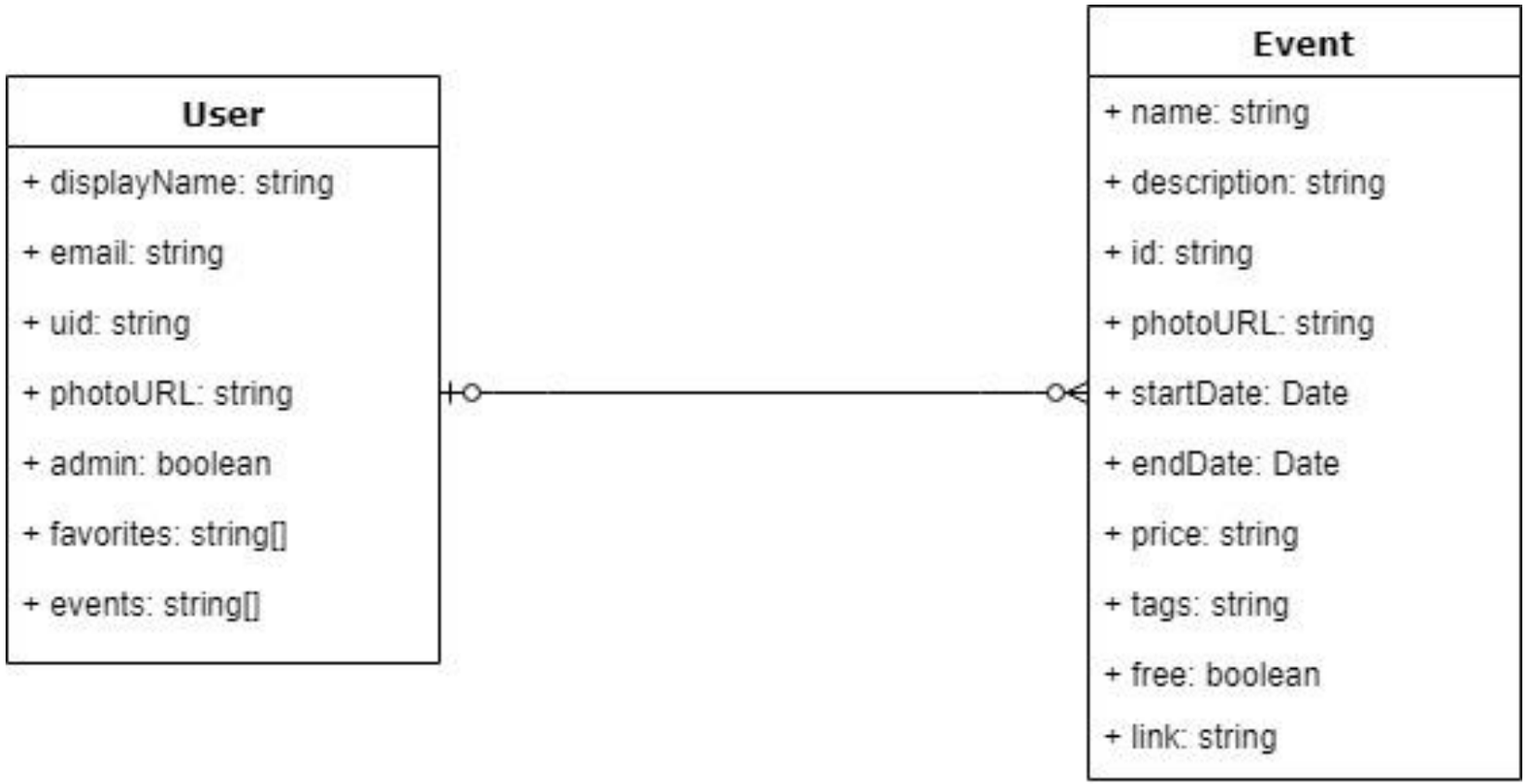
15. Firebase Authentication: simple, free multi-platform sign-in [Електронний ресурс]. — Режим доступу: https://goo.gl/dYf9nn. Дата доступу: травень 2017. Назва з екрану.

ДОДАТКИ

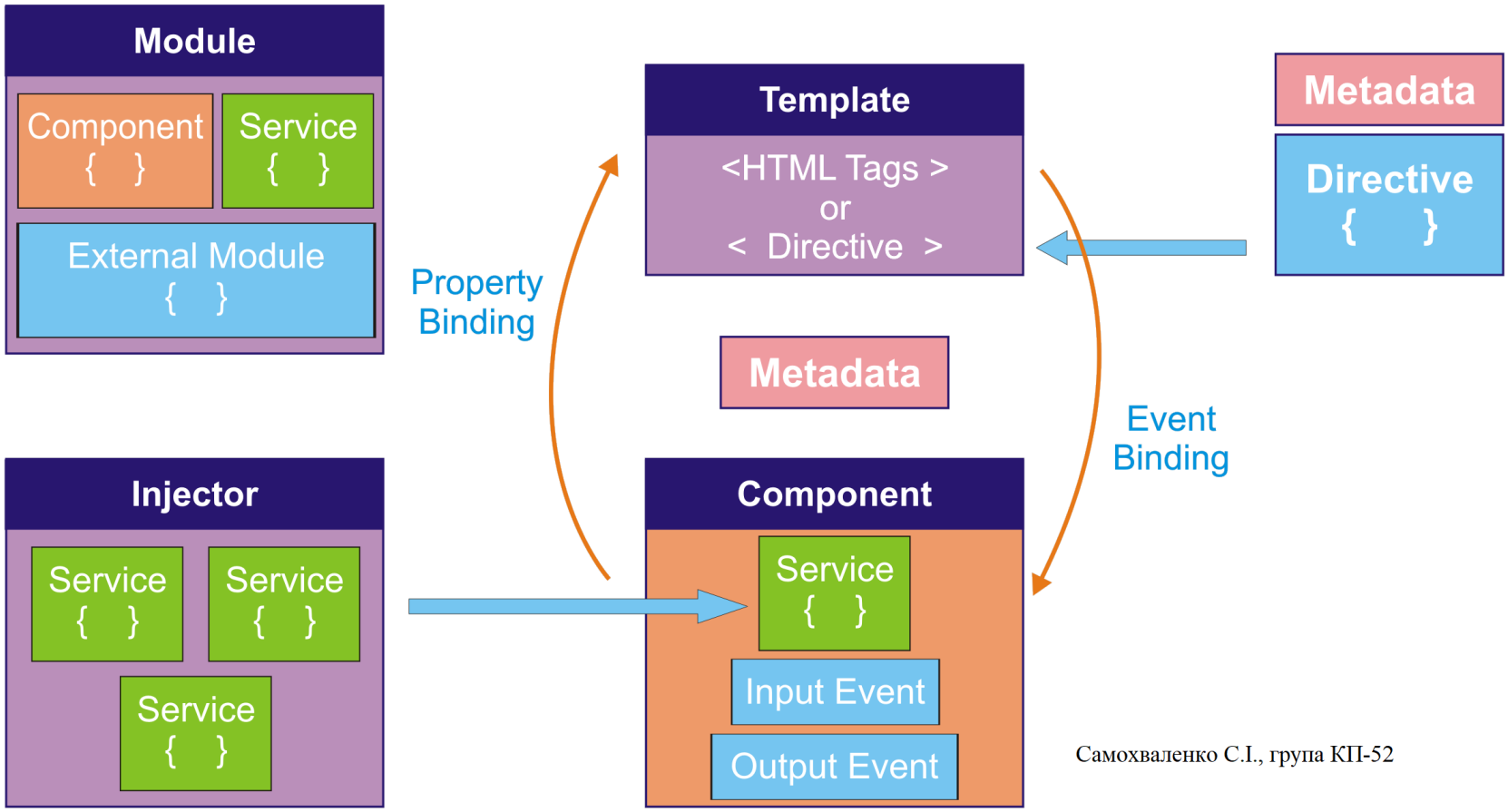
Додаток 1
Копії графічних матеріалів



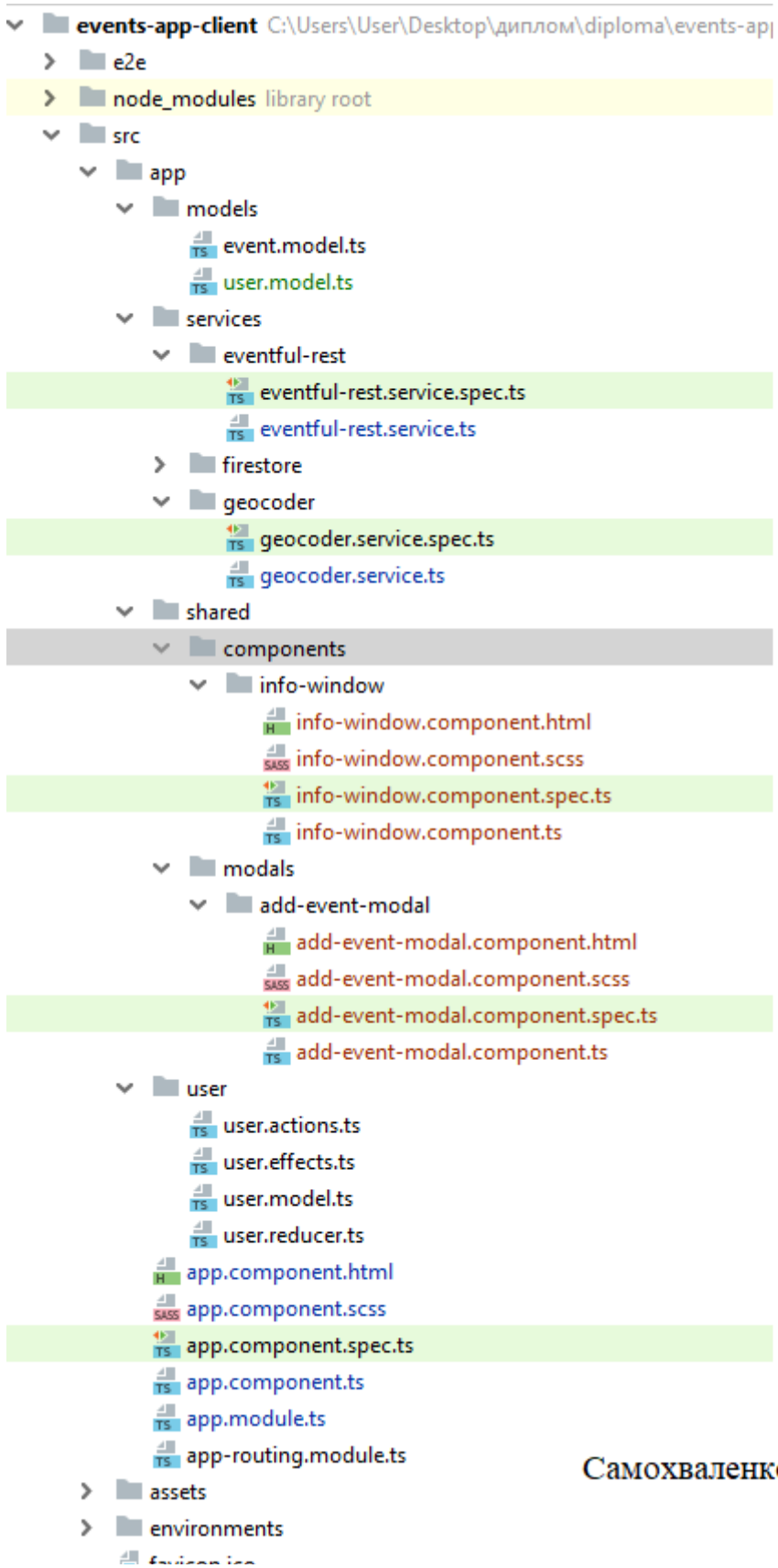
ДП. 045440-06-99
Веб-ресурс "Мапа подій".
Функціонал програмних засобів. Діаграма прецедентів



ДП.045440-07-99
Веб-ресурс "Мапа подій"
Структура бази даних. ERD діаграма



Самохваленко С.І., група КП-52



Самохваленко С.І., група КП-52

Додаток 2
Копія презентації

Веб-ресурс “Мапа подій”

Самохваленко Станіслав, КП-52
Науковий керівник: старший викладач каф. ПЗКС ФПМ,
Ю.В.Бухтіяров

Мета проекту

Створити веб-ресурс, на якому можна легко знайти та переглянути деталі про події, що цікавлять користувача, з можливістю фільтрації, сортування, пошуку даних, а також додавання власних подій.

Актуальність

Веб-ресурс “Мапа подій” для перегляну та пошуку подій, що цікавлять – це нове рішення, яке дозволить користувачам знаходити актуальну та потрібну інформацію про події, що цікавлять, та з допомогою нього планувати своє дозвілля у будь-якій точці світу.

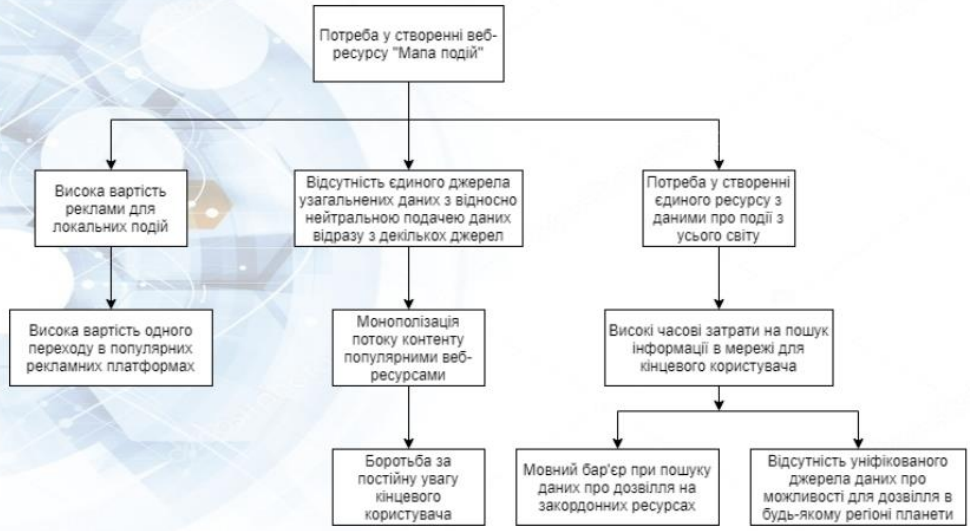
3/20

Проблеми, що вирішує проект

- Засилля так званого інформаційного сміття в мережі інтернет
- Велика кількість відволікаючих факторів при пошуку потрібного контенту
- Відсутність аналогів, або ж їхня вузька тематика
- Дозволить просувати свої події локальним організаторам
- Зниження кінцевої вартості реклами за рахунок рівних умов розміщення і відображення подій для користувачів

4/20

Дерево проблем



Існуючі рішення

KYIV MAPS - локальність та обмеженість даних

eventful - орієнтованість на розробників

CONCERT.UA - платформа для просування та продаж

Постановка задачі

Мета: забезпечити можливість перегляду інформації про події по всьому світу з можливістю фільтрації, пошуку та сортування наданої інформації, а також розміщення власних подій.

Завдання:

1. Проаналізувати існуючі програмні рішення
2. Розробити програмний застосунок відповідно до вимог
3. Протестувати роботу веб-додатку

7/20

Вимоги до продукту

- Забезпечення зручності, якості, швидкого розуміння роботи з програмою
- Можливість перегляду подій з усього світу на інтерактивній мапі
- Можливість фільтрації, пошуку та сортування подій.
- Можливість додавання власних подій на мапу для користувачів
- Можливість додавання подій в список улюблених для користувачів та підписка на нові події з профілів організаторів
- Монетизація: просування подій та виділення їх маркерів на мапі за додаткову оплату

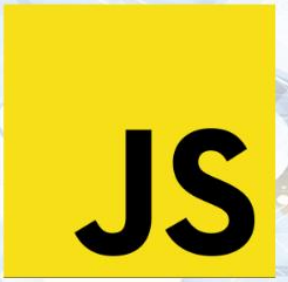
8/20

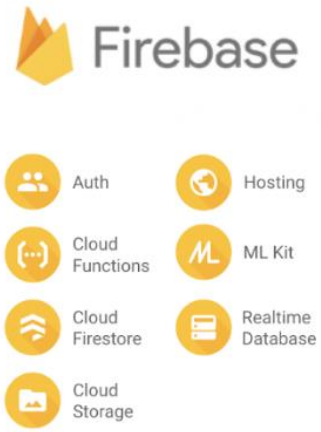
Вимоги до інтерфейсу

- Інтерфейс системи мати інтуїтивно зрозумілі інтерактивні елементи а також забезпечувати просту систему пошуку та фільтрації контенту.
- Елементи інтерфейсу повинні бути однаково зрозумілими користувачам з різних мовних регіонів.
- Інтерфейс системи повинен бути розрахований на користувачів, які не мають спеціальних технічних знань і навичок в області комп'ютерної техніки.

9/20

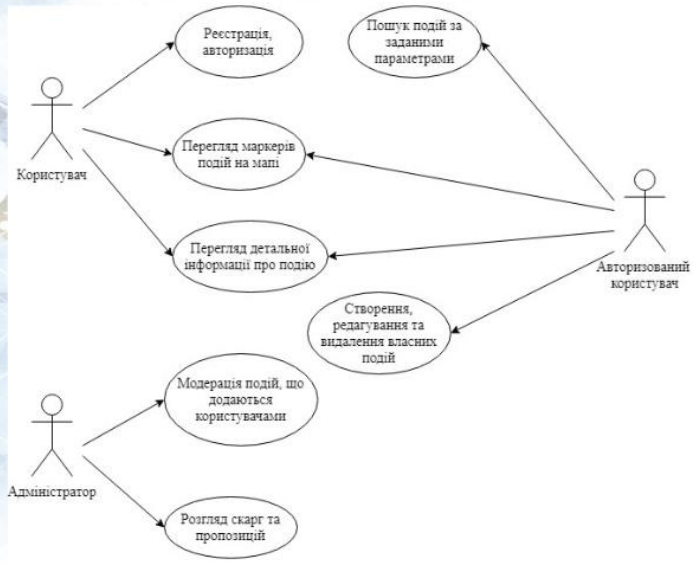
Вибір технологій



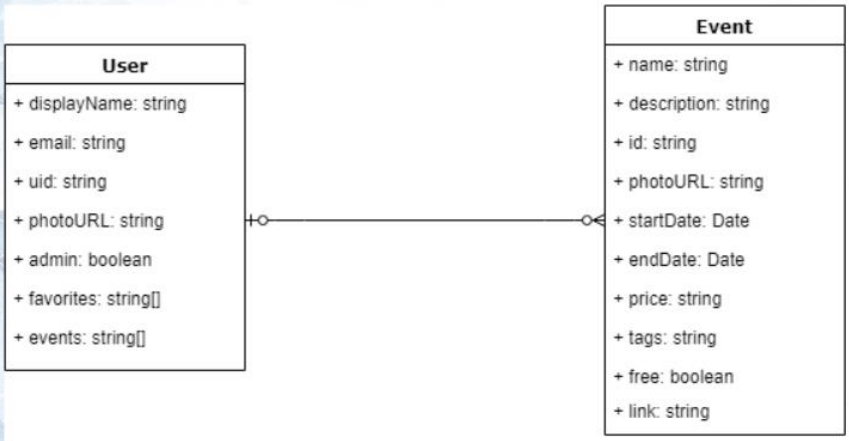


10/20

Діаграма прецедентів

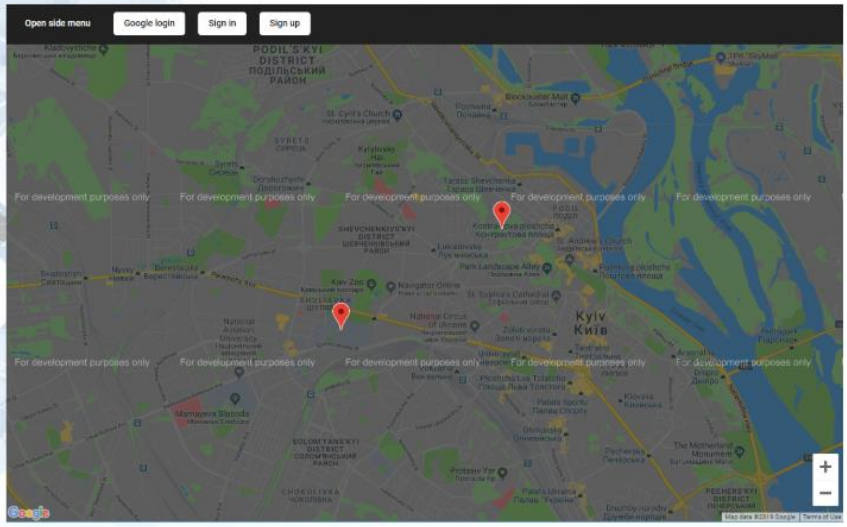


Діаграма структури СУБД



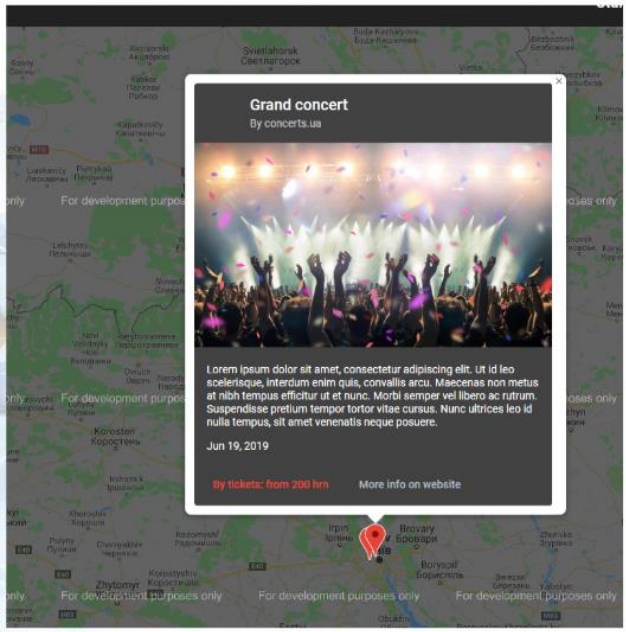
ДП.045440-07-99
Веб-ресурс "Мапа подій"
Структура бази даних. ERD діаграма

Приклади роботи



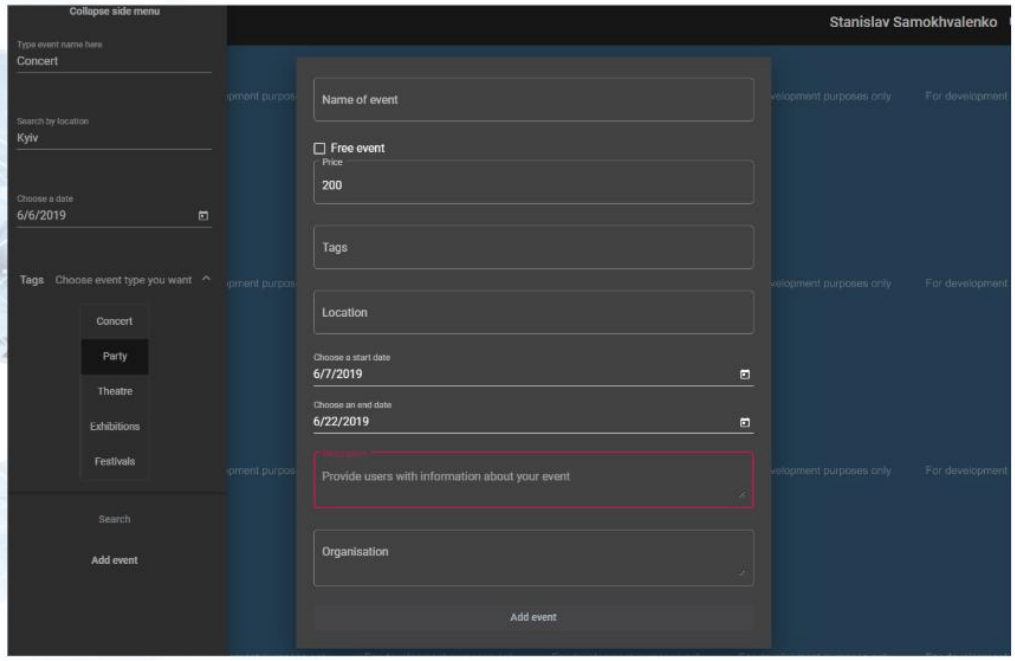
Вигляд головної сторінки для незареєстрованого користувача

13/20



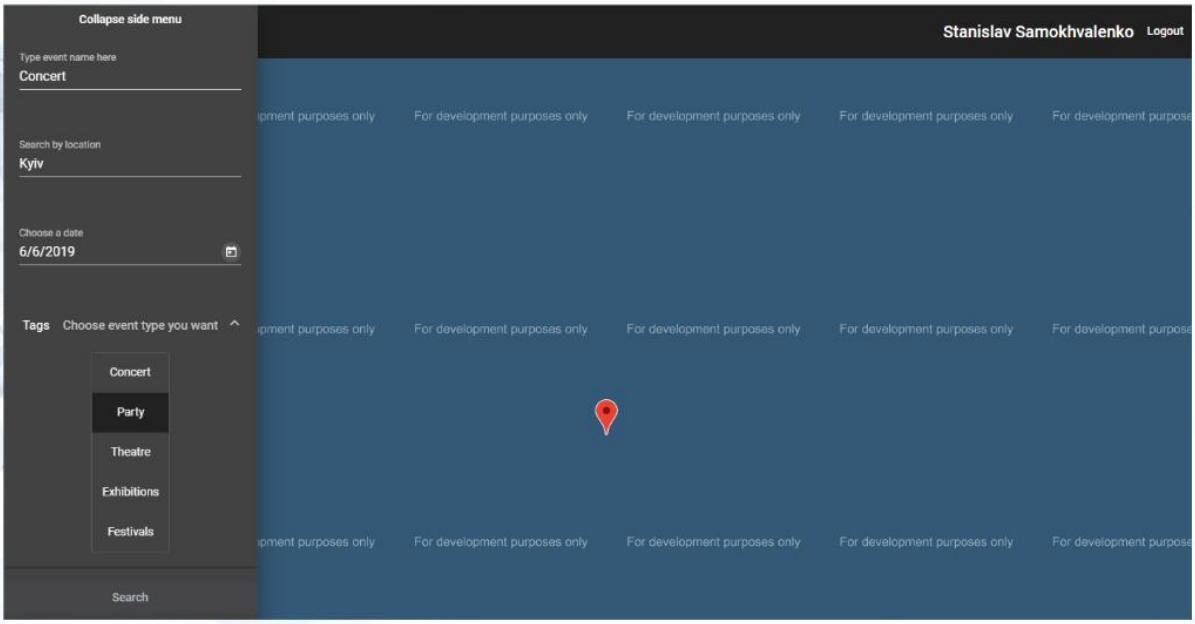
Інформація про подію в розгорнутому вигляді після кліку на маркер

14



Форма додавання події

15



Форма фільтрації подій в боковому меню

16

Висновки

- проведено аналіз доступних програмних рішень;
- розроблено структуру веб-додатку та архітектуру БД;
- реалізовано клієнтську частину;
- реалізовано вимоги до ПЗ
- веб-сайт у вигляді інтерактивної мапи з подіями
- протестовано роботу веб-додатку.

Дякую за увагу!

Додаток 3
Лістинг програми

Лістинг 1. Сервіс для геокодингу.

```
import {Injectable} from '@angular/core';
import {MapsAPILoader} from '@agm/core';
import {Observable, of, from} from 'rxjs';
import {map, switchMap, tap} from 'rxjs/operators';

declare var google: any;

@Injectable({
  providedIn: 'root'
})
export class GeocoderService {
  private geocoder: any;

  constructor(private mapLoader: MapsAPILoader) {
  }

  private initGeocoder() {
    this.geocoder = new google.maps.Geocoder();
  }

  private waitForMapsToLoad(): Observable<boolean> {
    if(!this.geocoder) {
      return from(this.mapLoader.load())
        .pipe(
          tap(() => this.initGeocoder()),
          map(() => true)
        );
    }
    return of(true);
  }

  geocodeAddress(location: string): Observable<any> {
    return this.waitForMapsToLoad().pipe(
      // filter(loaded => loaded),
      switchMap(() => {
        return new Observable((observer) => {
          this.geocoder.geocode({address: location}, (results, status) => {
            if (status === google.maps.GeocoderStatus.OK) {
              console.log('Geocoding complete!');
              observer.next({
                lat: results[0].geometry.location.lat(),
                lng: results[0].geometry.location.lng()
              });
            } else {
              console.log('Error - ', results, ' & Status - ', status);
              observer.next({ lat: 0, lng: 0 });
            }
            observer.complete();
          });
        });
      })
    );
  }
}
```

Лістинг 2. Кореневий модуль додатку

```
  @NgModule({
  declarations: [
    AppComponent,
    AddEventModalComponent
  ],
  imports: [
    BrowserModule,
    CommonModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    RouterModule,
    BrowserAnimationsModule,
    NoopAnimationsModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFirestoreModule,
    AngularFireAuthModule,
    AngularFireStorageModule,
    AgmCoreModule.forRoot({
      apiKey: 'AIzaSyDrhkCKbfCk8h0ujnU8EQH_sZJ0ia-1XrM',
    }),
    EffectsModule.forRoot([
      UserEffects
    ]),
    StoreModule.forRoot({
      user: userReducer
    }),
    AppRoutingModule,
    MatToolbarModule,
    MatSidenavModule,
    MatButtonModule,
    MatExpansionModule,
    MatTreeModule,
    MatButtonToggleModule,
    MatProgressSpinnerModule,
    MatFormFieldModule,
    MatInputModule,
    MatDialogModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatCardModule,
  ],
  entryComponents: [AddEventModalComponent],
  providers: [GeocoderService],
  bootstrap: [AppComponent],
})
export class AppModule {
}
```

Лістинг 3. Шаблон форми додавання події

```
<form [formGroup]="form">
  <mat-form-field [formControl]="form.controls['name']" appearance="outline">
```

```
    <mat-label>Name of event</mat-label>
    <input matInput placeholder="Type name of your event">
  </mat-form-field>
  <mat-checkbox>Free event</mat-checkbox>
  <mat-form-field [formControl]="form.controls['price']" appearance="outline">
    <mat-label>Price</mat-label>
    <input matInput placeholder="Type price of your event">
  </mat-form-field>
  <mat-form-field [formControl]="form.controls['tags']" appearance="outline">
    <mat-label>Tags</mat-label>
    <input matInput placeholder="Add tags that are relevant to your event separated
by space">
  </mat-form-field>
  <mat-form-field [formControl]="form.controls['startDate']">
    <input matInput [matDatepicker]="picker" placeholder="Choose a start date">
    <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
    <mat-datepicker #picker></mat-datepicker>
  </mat-form-field>
  <mat-form-field [formControl]="form.controls['endDate']">
    <input matInput [matDatepicker]="picker2" placeholder="Choose an end date">
    <mat-datepicker-toggle matSuffix [for]="picker2"></mat-datepicker-toggle>
    <mat-datepicker #picker2></mat-datepicker>
  </mat-form-field>
  <mat-form-field                    [formControl]="form.controls['description']"
appearance="outline">
    <mat-label>Description</mat-label>
    <textarea matInput placeholder="Provide users with information about your
event"></textarea>
  </mat-form-field>
  <mat-form-field                   [formControl]="form.controls['organisation']"
appearance="outline">
    <mat-label>Organisation</mat-label>
    <textarea matInput placeholder="Provide some information about you or your
organisation"></textarea>
  </mat-form-field>
  <button mat-button color="accent" type="submit">Add event</button>
</form>
```

Лістинг 4. Сервіс для зв’язку з базою даних

```
      @Injectable({
  providedIn: 'root',
})
export class EventService {

  private _done = new BehaviorSubject(false);
  public done = new Observable<boolean>();
  private _loading = new BehaviorSubject(false);
  private prev: boolean;
  private _data = new BehaviorSubject([]);
  private query: QueryConfig;
  private counter: number;
  public nextPageExists: boolean;
  public data: Observable<Event[]>;

  userId: string;
```

```
  constructor(private afs: AngularFirestore, private afAuth: AngularFireAuth,
private router: Router) {
    this.afAuth.authState.subscribe(user => {
      if (user) {
        this.userId = user.uid;
      }
    });
  }

  init(query) {
    this._done.next(false);
    this.prev = false;
    this.counter = 0;
    this.query = query;

    const first = this.afs.collection(this.query.path, ref => {
      return this.query.whereFilter ?
        ref
        .where(this.query.whereField, '==', this.query.whereFiledValue)
        .orderBy(this.query.orderByField, this.query.descending ? 'desc' : 'asc')
        .limit(this.query.limit) :
        ref
        .orderBy(this.query.orderByField, this.query.descending ? 'desc' : 'asc')
        .limit(this.query.limit);
    });

    this.mapAndUpdate(first);

    // Create the observable array for consumption in components
    this.data = this._data.asObservable()
      .scan( (acc, val) => {
        return val;
      });
    this.done = this._done.asObservable()
      .scan( (acc, val) => {
        return acc;
      });
  }

  // Retrieves additional data from firestore
  page() {
    const cursor = this.getCursor(this.query.descending);
    const more = this.afs.collection(this.query.path, ref => {
      return this.query.whereFilter ?
        ref
        .where(this.query.whereField, '==', this.query.whereFiledValue)
        .orderBy(this.query.orderByField, this.query.descending ? 'desc' : 'asc')
        .limit(this.query.limit)
        .startAfter(cursor) :
        this.query.previous ?
          ref
          .orderBy(this.query.orderByField,  this.query.descending  ?  'desc'  :
'asc')
          .limit(this.query.limit)
          .startAt(cursor) :
          ref
            .orderBy(this.query.orderByField,  this.query.descending  ?  'desc'  :
'asc')
```

```
            .limit(this.query.limit)
            .startAfter(cursor);
    });
    this.mapAndUpdate(more);
    this.query.previous ? this.counter-- : this.counter++;
  }

  setPrev(val) {
    this.query.previous = val;
  }

  setDesc(val) {
    this.query.descending = val;
  }

  public GetPaginateCounter(): number {
    return this.counter;
  }

  // Determines the doc snapshot to paginate query
  private getCursor(direction) {
    const current = this._data.value;
    if (current.length) {
      return direction ? current[current.length - 1].doc : current[0].doc;
    }
    return null;
  }

  // Maps the snapshot to usable format the updates source
  private mapAndUpdate(col: AngularFirestoreCollection<any>) {

    if (this._done.value && !this.query.previous) {
      return;
    }
    // loading
    this._loading.next(true);

    // Map snapshot with doc ref (needed for cursor)
    return col.snapshotChanges()
      .do(arr => {
        let values = arr.map(snap => {
          const data = snap.payload.doc.data();
          const doc = snap.payload.doc;
          return { ...data, doc };
        });
        if (values.length === 0 && this.counter === 0) {
          this.router.navigate(['/notfound']);
        }

        if (values.length < 10) {
          this.nextPageExists = false;
        } else {
          this.nextPageExists = true;
        }

        this.query.previous ? values = values.slice(1, 10) : values =
values.slice(0, 9);
```

```
        values = this.query.previous ? values.reverse() : values;
        // update source with new values, done loading
        this._data.next(values);
        this._loading.next(false);

        // no nextPage values, mark done
      })
      .take(8)
      .subscribe();
  }

  getEventsByUser(user): Observable<Event[]> {
    return this.afs.collection< Event >('events',
        ref => ref
      .where('user', '==', user)
          .limit(10))
      .valueChanges();
  }

  getEventsByTag(tag): Observable<Event[]> {
    return  this.afs.collection<Event>('events',  ref  =>  ref.where('tag',  '==',
tag)).valueChanges();
  }

  getEventById(Id): Observable<Article> {
    return this.afs.doc<Event>(`events/${id}`).valueChanges();
  }

  createEvent(item: Event) {
    this.updateTag(item.tag);
    this.afs.doc<Event>(`events/${item.id}`).set(item);
  }

  updateEvent(item: Event) {
    this.afs.doc<Event>(`events/${item.id}`).update(item);
  }

  getEventsByTag(tag): Observable<Event[]> {
    return  this.afs.collection<Event>('events',  ref  =>  ref.where(tag,  '==',
true)).valueChanges();
  }
  }
}
```

Лістинг 5. Шаблон головної сторінки

```
    <mat-sidenav-container
  class="example-container" (backdropClick)="close()">
  <mat-sidenav #sidenav (keydown.escape)="close('escape')">
    <button mat-button (click)="close()">Collapse side menu</button>
    <mat-form-field>
      <input matInput placeholder="Type event name here" value="Concert">
    </mat-form-field>
    <mat-form-field>
      <input matInput [matDatepicker]="picker" placeholder="Choose a date">
      <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
      <mat-datepicker #picker></mat-datepicker>
    </mat-form-field>
    <mat-accordion>
```

```
      <mat-expansion-panel>
        <mat-expansion-panel-header>
          <mat-panel-title>
            Tags
          </mat-panel-title>
          <mat-panel-description>
            Choose event type you want
          </mat-panel-description>
        </mat-expansion-panel-header>
        <mat-button-toggle-group name="fontStyle" aria-label="Font Style">
          <mat-button-toggle  (click)="toggleTag('concert')">Concert</mat-button-
toggle>
          <mat-button-toggle      (click)="toggleTag('party')">Party</mat-button-
toggle>
          <mat-button-toggle  (click)="toggleTag('theatre')">Theatre</mat-button-
toggle>
          <mat-button-toggle (click)="toggleTag('exhibitions')">Exhibitions</mat-
button-toggle>
          <mat-button-toggle     (click)="toggleTag('festivals')">Festivals</mat-
button-toggle>
        </mat-button-toggle-group>
      </mat-expansion-panel>
    </mat-accordion>
    <button mat-button (click)="addEvent()">Add event</button>
  </mat-sidenav>

  <mat-sidenav-content>
    <mat-toolbar>
      <button mat-button (click)="sidenav.open()">Open side menu</button>
      <ng-container *ngIf="user$ | async as user">
        <button  mat-button  class="google-button"  *ngIf="user.uid  ==  null"
(click)="googleLogin()">Google login</button>
        <button  mat-button  class="google-button"  *ngIf="user.uid  ==  null"
(click)="signIn()">Sign in</button>
        <button  mat-button  class="google-button"  *ngIf="user.uid  ==  null"
(click)="signUp()">Sign up</button>

      </ng-container>
      <span class="spacer"></span>
      <ng-container *ngIf="user$ | async as user">
        <div *ngIf="user.uid != null">{{user.displayName}}</div>
        <button        *ngIf="user.uid        !=        null"        mat-button
(click)="logout()">Logout</button>
      </ng-container>
    </mat-toolbar>
    <agm-map
      [latitude]="locations[0]?.lat"
      [longitude]="locations[0]?.lng"
      [zoom]="8">
      <agm-marker *ngFor="let event of events"
                  [latitude]="event.location?.lat"
                  [longitude]="event.location?.lng">
        <agm-info-window [maxWidth]="500">
          <mat-card class="example-card">
            <mat-card-header>
              <div mat-card-avatar class="example-header-image"></div>
              <mat-card-title>{{event.name}}</mat-card-title>
              <mat-card-subtitle>{{event.organizer}}</mat-card-subtitle>
```

```
            </mat-card-header>
            <img mat-card-image [src]="event.photoURL" alt="Photo of a Shiba Inu">
            <mat-card-content>
              <p>
                {{event.}}
              </p>
              <div class="date">{{date | date}}</div>
            </mat-card-content>
            <mat-card-actions>
              <button      mat-button      color="warn">By      tickets:      from
{{event.price}}</button>
              <a mat-button color="accent" [routerLink]="['/' + event.url]">More
info on website</a>
            </mat-card-actions>
          </mat-card>
        </agm-info-window>
      </agm-marker>
    </agm-map>
    <p *ngIf="loading">Loading...</p>
  </mat-sidenav-content>
</mat-sidenav-container>
```

Лістинг 6. Сервіс пошуку подій

```
      @Injectable({
  providedIn: 'root'
})
export class EventfulRestService {

  constructor(private http: HttpClient) {
  }

  searchEvents(searchParams: string) {
    const queryUrl = `http://api.eventful.com/rest/events/search?${searchParams}`;
    return this.http.get(queryUrl);
  }
}
```

Лістинг 7. Модуль авторизації та реєстрації

```
@Injectable()
export class UserEffects {

  constructor(private actions: Actions, private afAuth: AngularFireAuth) {
  }

  @Effect()
  getUser: Observable<Action> = this.actions.pipe(ofType(userActions.GET_USER),
    map((action: userActions.GetUser) => action.payload),
    switchMap(payload => this.afAuth.authState),
    map((authData: User) => {
      if (authData) {
        /// User logged in
        const user = new User(authData.uid, authData.displayName, authData.email);
        return new userActions.Authenticated(user);
      } else {
        /// User not logged in
```

```
        return new userActions.NotAuthenticated();
      }

    }),
    catchError((err: any) => of(new userActions.AuthError()))
  );

  @Effect()
  login: Observable<Action> = this.actions.pipe(
    ofType(userActions.GOOGLE_LOGIN),
    map((action: userActions.GoogleLogin) => action.payload),
    switchMap(payload => {
      return this.googleLogin();
    }),
    map( credential => {
      // successful login
      return new userActions.GetUser();
    }),
    catchError(err => {
      return of(new userActions.AuthError({error: err.message}));
    }));

  @Effect()
  logout: Observable<Action> = this.actions.pipe(
    ofType(userActions.LOGOUT),
    map((action: userActions.Logout) => action.payload ),
    switchMap(payload => {
      return from( this.afAuth.auth.signOut());
    }),
    map( authData => {
      return new userActions.NotAuthenticated();
    }),
    catchError(err => of(new userActions.AuthError({error: err.message})) )
  );


  private googleLogin(): Observable<any> {
    const provider = new firebase.auth.GoogleAuthProvider();
    return of(this.afAuth.auth.signInWithPopup(provider));
  }
}

import { Action } from '@ngrx/store';
import { User } from './user.model';

export const GET_USER               = '[Auth] Get user';
export const AUTHENTICATED          = '[Auth] Authenticated';
export const NOT_AUTHENTICATED      = '[Auth] Not Authenticated';

export const GOOGLE_LOGIN           = '[Auth] Google login attempt';
export const LOGOUT                 = '[Auth] Logout';

export const AUTH_ERROR             = '[Auth] Error';


/// Get User AuthState

export class GetUser implements Action {
```

```
  readonly type = GET_USER;
  constructor(public payload?: any) {}
}

export class Authenticated implements Action {
  readonly type = AUTHENTICATED;
  constructor(public payload?: any) {}
}

export class NotAuthenticated implements Action {
  readonly type = NOT_AUTHENTICATED;
  constructor(public payload?: any) {}
}

export class AuthError implements Action {
  readonly type = AUTH_ERROR;
  constructor(public payload?: any) {}
}

/// Google Login Actions

export class GoogleLogin implements Action {
  readonly type = GOOGLE_LOGIN;
  constructor(public payload?: any) {}
}

/// Logout Actions

export class Logout implements Action {
  readonly type = LOGOUT;
  constructor(public payload?: any) {}
}

export type All
  = GetUser
  | Authenticated
  | NotAuthenticated
  | GoogleLogin
  | AuthError
  | Logout;


import * as userActions from './user.actions';
import { User } from './user.model';

export type Action = userActions.All;

const defaultUser = new User(null, 'GUEST', null);

/// Reducer function
export function userReducer(state: User = defaultUser, action: Action) {
  switch (action.type) {

    case userActions.GET_USER:
      return { ...state, loading: true };

    case userActions.AUTHENTICATED:
      return { ...state, ...action.payload, loading: false };
```

```
    case userActions.NOT_AUTHENTICATED:
      return { ...state, ...defaultUser, loading: false };

    case userActions.GOOGLE_LOGIN:
      return { ...state, loading: true };

    case userActions.AUTH_ERROR:
      return { ...state, ...action.payload, loading: false };

    case userActions.LOGOUT:
      return { ...state, loading: true };

  }
}
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-РЕСУРС «МАПА ПОДІЙ»

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ С.І. Самохваленко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-ресурс «Мапа Подій», написаний на мові програмування JavaScript з використанням фреймворку Angular.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Функціональна працездатність елементів сторінок web-сайту.
2. Наявність доступу до бази даних подій.
3. Взаємодію сервера з клієнтом.
4. Забезпечення коректної обробки запитів від користувача.
5. Забезпечення належного рівня безпеки даних.
6. Зручність роботи з web-сайтом.
7. Відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом функціонального тестування. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні Critical path test (базове тестування).
2. Тестування продуктивності програмного забезпечення, зокрема Performance testing (тестування стабільності).
3. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування web-ресурсу в різних web-браузерах.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування зручності використання.
8. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А.Дичка

"___" _____ 2019 р.

ВЕБ-РЕСУРС «МАПА ПОДІЙ»

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Ю.В. Бухтіяров

Нормоконтроль:

_____ М.В.Онай

Виконавець:

_____С.І. Самохваленко

2019

1. Опис структури додатку

Інформаційна система представлена односторінковим веб-додатком, який складається з динамічних компонентів, які об’єднані у модулі. Компоненти за допомогою сервісів взаємодіють із сервером. Основне призначення – перегляд, пошук та фільтрація подій на інтерактивній мапі.

Застосунок включає наступні компоненти:

– Інтерактивна карта;

– хедер з можливостями для авторизації;

– бокове меню з фільтрами та списком улюблених подій;

– модальні вікна для додавання та редагування подій.

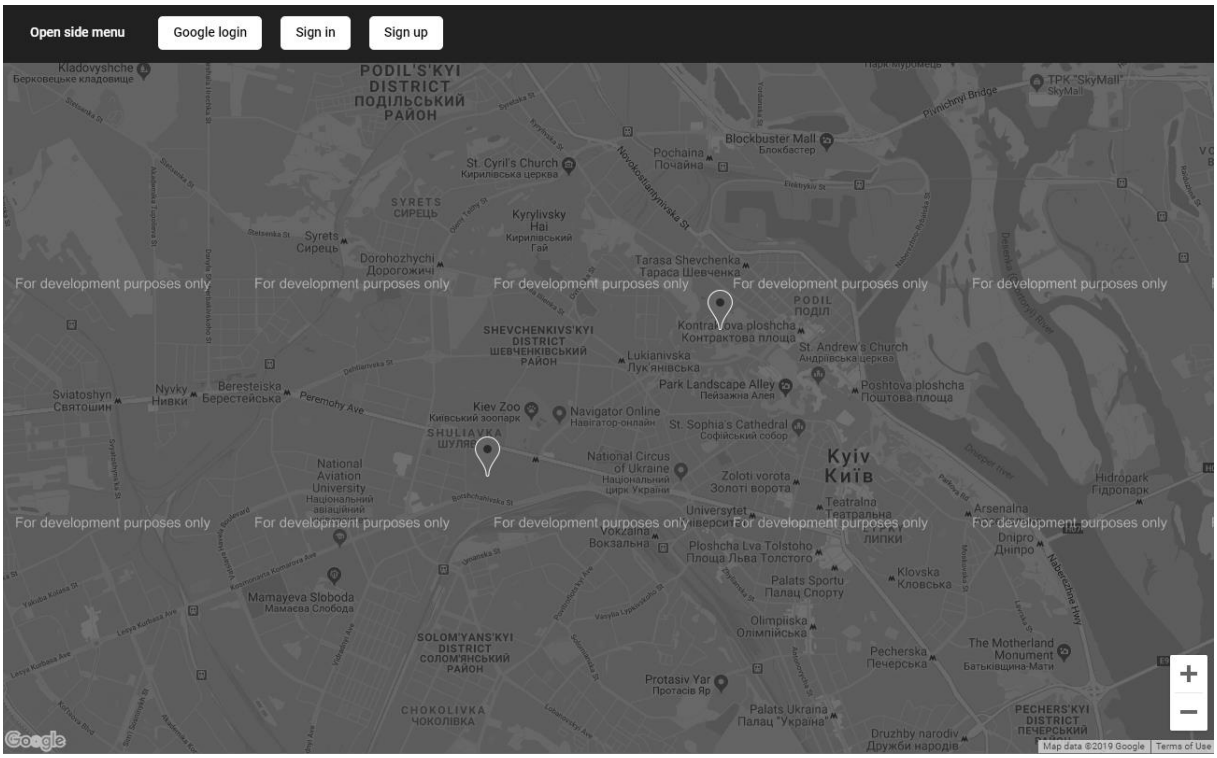


Рис. 1. «Мар», вигляд для неавторизованого користувача

Рис. 2. Детальне відображення інформації про подію

2. Опис вмісту веб-сторінок

Додаток містить всю інформацію на одній сторінці та має верхнє меню для швидкого доступу до всіх функціональних можливостей. Кожен маркер є інтерактивним, після кліку по одному з них відкривається інформаційне вікно з деталями про подію (рис. 2). Після авторизації стає доступним бокове меню з фільтрами та інтерфейсом для додавання подій (рис. 3).

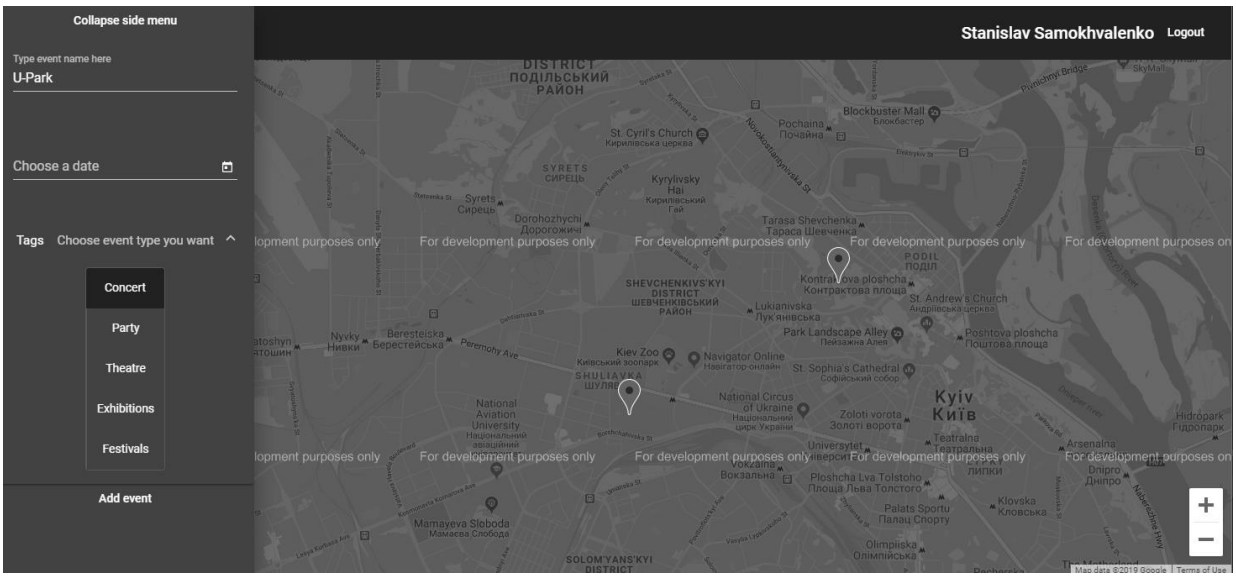


Рис. 3. Бокова панель з інструментами

3. Процедура додавання події

Користувач може додати подію після авторизації, натиснувши відповідну кнопку. Для незареєстрованих користувачів дана можливість відсутня.

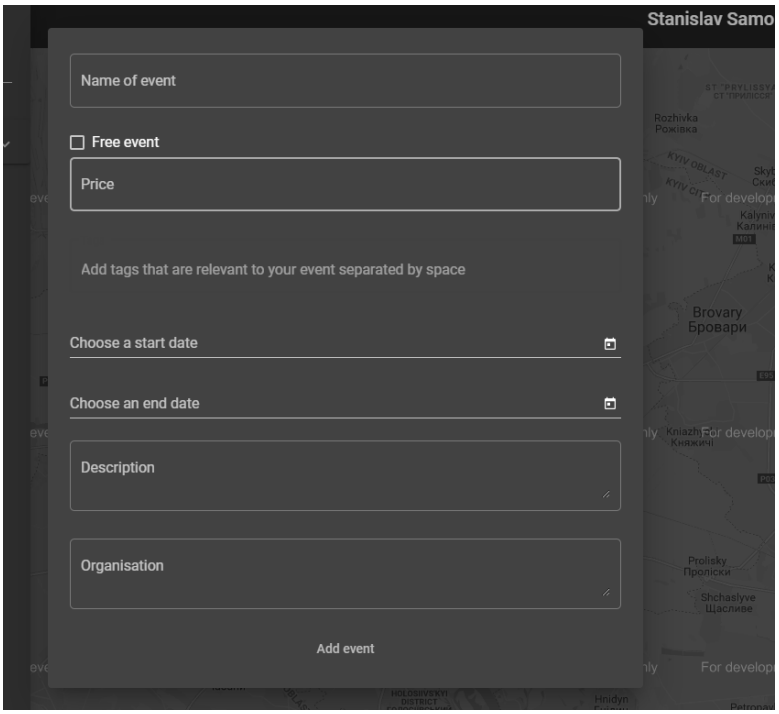


Рис. 4. Модальне вікно створення події